Polygonal Number Theorem

Kevin Lee, Zhengkun Ye and Angeliki Koutsoukou-Argyraki

September 13, 2023

Abstract

We formalize the proofs of Cauchy's and Legendre's Polygonal Number Theorems given in Melvyn B. Nathanson's book 'Additive Number Theory: The Classical Bases' [2].

For $m \geq 1$, the k-th polygonal number of order $m+2$ is defined to be $p_m(k) = \frac{mk(k-1)}{2} + k$. The theorems state that:

- If $m \geq 4$ and $N \geq 108m$, then N can be written as the sum of $m+1$ polygonal numbers of order $m+2$, at most four of which are different from 0 or 1. If $N \geq 324$, then N can be written as the sum of five pentagonal numbers, at least one of which is 0 or 1.
- Let $m \geq 3$ and $N \geq 28m^3$. If m is odd, then N is the sum of four polygonal numbers of order $m+2$. If m is even, then N is the sum of five polygonal numbers of order $m+2$, at least one of which is 0 or 1.

We also formalize the proof of Gauss's theorem which states that every non-negative integer is the sum of three triangular numbers.

Contents

1 Technical Lemmas **3**
1.1 Lemma 1.10 in [2] 3
1.2 Lemma 1.11 in [2] 7
1.3 Lemma 1.12 in [2] 10

2 Polygonal Number Theorem **20**
2.1 Gauss's Theorem on Triangular Numbers 20
2.2 Cauchy's Polygonal Number Theorem 21
2.3 Legendre's Polygonal Number Theorem 39

Acknowledgements

The project was completed during the 2023 summer internship of the first two authors within the Cambridge Mathematics Placements (CMP) Programme, supervised by the third author and hosted at the Department of Computer Science and Technology, University of Cambridge. All three authors were funded by the ERC Advanced Grant ALEXANDRIA (Project GA 742178) led by Lawrence C. Paulson.

Kevin Lee and Zhengkun Ye wish to thank the Zulip community for help with beginners' questions.

1 Technical Lemmas

We show three lemmas used in the proof of both main theorems.

theory *Polygonal-Number-Theorem-Lemmas*
imports *Three-Squares.Three-Squares*

begin

1.1 Lemma 1.10 in [2]

This lemma is split into two parts. We modify the proof given in [2] slightly as we require the second result to hold for $l = 2$ in the proof of Legendre's polygonal number theorem.

theorem *interval-length-greater-than-four*:
fixes *m N L* :: *real*
assumes $m \geq 3$
assumes $N \geq 2*m$
assumes $L = (2/3 + sqrt\ (8*N/m - 8)) - (1/2 + sqrt\ (6*N/m - 3))$
shows $N \geq 108*m \Longrightarrow L > 4$

proof −
assume *asm*: $N \geq 108*m$
show $L > 4$
proof −
define *x* :: *real* **where** $x = N\ /\ m$
define *l* :: *real* **where** $l = 4$
define *l-0* :: *real* **where** $l\text{-}0 = 4 - 1/6$
have *0*: $x \geq 2$ **unfolding** *x-def* **using** *assms*(*2*)
by (*metis assms*(*1*) *divide-right-mono dual-order.trans linorder-le-cases mult.commute mult-1 nonzero-mult-div-cancel-left numeral-le-one-iff semiring-norm*(*70*) *zero-le-square*)
have *1*: $L = sqrt\ (8*x - 8) - sqrt\ (6*x - 3) + 1/6$ **by** (*auto simp add*: *x-def assms*(*3*))
hence *2*: $L > l \longleftrightarrow sqrt\ (8*x - 8) > sqrt\ (6*x - 3) + l\text{-}0$ **unfolding** *l-0-def l-def* **by** *auto*
have *3*: $sqrt\ (8*x - 8) > sqrt\ (6*x - 3) + l\text{-}0 \longleftrightarrow 2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3)$
proof
assume $sqrt\ (8*x - 8) > sqrt\ (6*x - 3) + l\text{-}0$
hence $(sqrt\ (8*x - 8))\hat{}2 > (sqrt\ (6*x - 3) + l\text{-}0)\hat{}2$
using *l-0-def asm* **by** (*smt* (*verit*, *ccfv-SIG*) *0 divide-less-eq-1-pos one-power2 pos2 power-mono-iff real-less-rsqrt*)
hence $8*x - 8 > 6*x - 3 + l\text{-}0\hat{}2 + 2*l\text{-}0 * sqrt\ (6*x - 3)$
by (*smt* (*verit*, *del-insts*) *0 power2-sum real-sqrt-pow2-iff*)
thus $2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3)$ **by** *auto*
next
assume $2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3)$
hence $8*x - 8 > 6*x - 3 + l\text{-}0\hat{}2 + 2*l\text{-}0 * sqrt\ (6*x - 3)$ **by** *auto*

hence (*sqrt* (*8*x − 8*))^2 > (*sqrt* (*6*x − 3*) + *l-0*)^2
by (*smt* (*verit*, *best*) *0 power2-sum real-sqrt-pow2-iff*)
thus *sqrt* (*8*x − 8*) > *sqrt* (*6*x − 3*) + *l-0*
using *0 real-less-rsqrt* **by** *force*
qed
have *2*x − l-0^2 − 5 > 2*l-0* sqrt* (*6*x − 3*) ⟷ *4*x*(x − (*7*l-0^2 + 5*)) + (*l-0^2 + 5*)^2 + 12*l-0^2 > 0*
proof
assume *2*x − l-0^2 − 5 > 2*l-0* sqrt* (*6*x − 3*)
hence (*2*x − l-0^2 − 5*)^2 > (*2*l-0* sqrt* (*6*x − 3*))^2
by (*smt* (*verit*, *del-insts*) *0 asm l-0-def le-divide-eq-1-pos less-1-mult one-power2 pos2 power-mono-iff sqrt-le-D*)
thus *4*x*(x − (*7*l-0^2 + 5*)) + (*l-0^2 + 5*)^2 + 12*l-0^2 > 0*
using *0* **by** (*simp add*: *algebra-simps power2-eq-square power4-eq-xxxx*)
next
assume *4*x*(x − (*7*l-0^2 + 5*)) + (*l-0^2 + 5*)^2 + 12*l-0^2 > 0*
hence (*2*x − l-0^2 − 5*)^2 > (*2*l-0* sqrt* (*6*x − 3*))^2
using *0* **by** (*simp add*: *algebra-simps power2-eq-square power4-eq-xxxx*)
from *assms*(*1*) **have** *m > 0* **by** *auto*
hence *2*x ≥ 2*108*
using *x-def asm* **by** (*simp add*: *le-divide-eq*)
hence *2*x − l-0^2 − 5 ≥ 2*108 − (4−1/6)*(4−1/6) − 5* **unfolding** *l-0-def* **by** (*auto simp add*: *power2-eq-square*)
hence *2*x − l-0^2 − 5 > 0* **by** *auto*
thus *2*x − l-0^2 − 5 > 2*l-0* sqrt* (*6*x − 3*)
using ‹(*2*x − l-0^2 − 5*)^2 > (*2*l-0* sqrt* (*6*x − 3*))^2› **using** *power2-less-imp-less* **by** *fastforce*
qed
from *assms*(*1*) **have** *m > 0* **by** *auto*
hence *x ≥ 108* **using** *x-def asm* **by** (*simp add*: *le-divide-eq*)
have *7*(4−1/6)*(4−1/6) + 5 < (108*::*real*) **by** *simp*
hence *7*l-0^2 + 5 < 108* **unfolding** *l-0-def* **by** (*auto simp add*: *power2-eq-square*)
hence *x ≥ 7*l-0^2 + 5* **using** ‹*108 ≤ x*› **by** *auto*
hence *4*x*(x − (*7*l-0^2 + 5*)) + (*l-0^2 + 5*)^2 + 12*l-0^2 > 0*
by (*smt* (*verit*) *mult-nonneg-nonneg power2-less-eq-zero-iff zero-le-power2*)
thus *?thesis*
using *2 3* ‹(*2 * l-0 * sqrt* (*6 * x − 3*) < 2 * x − l-0^2 − 5*) = (*0 < 4 * x * (x − (7 * l-0^2 + 5*)) + (*l-0^2 + 5*)^2 + 12 * l-0^2*)› *l-def* **by** *blast*
qed
qed

theorem *interval-length-greater-than-lm*:
fixes *m N* :: *real*
fixes *L l* :: *real*
assumes *m ≥ 3*
assumes *N ≥ 2*m*
assumes *L = (2/3 + sqrt* (*8*N/m − 8*)) − (*1/2 + sqrt* (*6*N/m − 3*))*
shows *l ≥ 2 ∧ N ≥ 7*l^2*m^3 ⟹ L > l*m*

proof −
assume *asm*: $l \geq 2 \wedge N \geq 7*l\hat{}2*m\hat{}3$
show $L > l*m$
proof −
from *asm* **have** *asm1*: $l \geq 2$ **and** *asm2*: $N \geq 7*l\hat{}2*m\hat{}3$ **by** *auto*
define $x :: real$ **where** $x = N / m$
define *l-0* :: *real* **where** $l\text{-}0 = l*m - 1/6$
have $l\text{-}0 > 0$ **using** *l-0-def asm1 assms*(*1*)
by (*smt* (*verit*, *ccfv-threshold*) *le-divide-eq-1 mult-le-cancel-left2 of-int-le-1-iff*)
have *0*: $x \geq 2$ **using** *x-def assms*(*1*,*2*) **by** (*simp add*: *pos-le-divide-eq*)
have *1*: $L = sqrt\ (8*x - 8) - sqrt\ (6*x - 3) + 1/6$ **by** (*auto simp add*: *x-def assms*(*3*))
hence *2*: $L > l*m \longleftrightarrow sqrt\ (8*x - 8) > sqrt\ (6*x - 3) + l\text{-}0$ **by** (*auto simp add*: *l-0-def*)
have *3*: $sqrt\ (8*x - 8) > sqrt\ (6*x - 3) + l\text{-}0 \longleftrightarrow 2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3)$
proof
assume $sqrt\ (8*x - 8) > sqrt\ (6*x - 3) + l\text{-}0$
hence $(sqrt\ (8*x - 8))\hat{}2 > (sqrt\ (6*x - 3) + l\text{-}0)\hat{}2$
using *l-0-def asm1* **by** (*smt* (*verit*, *best*) ‹$0 < l\text{-}0$› *real-le-lsqrt real-sqrt-four real-sqrt-less-iff real-sqrt-pow2-iff*)
hence $8*x - 8 > 6*x - 3 + l\text{-}0\hat{}2 + 2* l\text{-}0 * sqrt\ (6*x - 3)$
by (*smt* (*verit*, *del-insts*) *0 power2-sum real-sqrt-pow2-iff*)
thus $2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3)$ **by** *auto*
next
assume $2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3)$
hence $8*x - 8 > 6*x - 3 + l\text{-}0\hat{}2 + 2*l\text{-}0 * sqrt\ (6*x - 3)$ **by** *auto*
hence $(sqrt\ (8*x - 8))\hat{}2 > (sqrt\ (6*x - 3) + l\text{-}0)\hat{}2$
by (*smt* (*verit*, *del-insts*) *0 power2-sum real-sqrt-pow2-iff*)
thus $sqrt\ (8*x - 8) > sqrt\ (6*x - 3) + l\text{-}0$
using *0 real-less-rsqrt* **by** *force*
qed
have $2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3) \longleftrightarrow 4*x*(x - (7*l\text{-}0\hat{}2 + 5)) + (l\text{-}0\hat{}2 + 5)\hat{}2 + 12*l\text{-}0\hat{}2 > 0$
proof
assume $2*x - l\text{-}0\hat{}2 - 5 > 2*l\text{-}0 * sqrt\ (6*x - 3)$
have $(2*x - l\text{-}0\hat{}2 - 5)\hat{}2 > (2*l\text{-}0 * sqrt\ (6*x - 3))\hat{}2$
using ‹$0 < l\text{-}0$› **by** (*smt* (*verit*, *ccfv-SIG*) *0* ‹$2 * l\text{-}0 * sqrt\ (6 * x - 3) < 2 * x - l\text{-}0^2 - 5$› *pos2 power-strict-mono real-sqrt-ge-zero zero-le-mult-iff*)
thus $4*x*(x - (7*l\text{-}0\hat{}2 + 5)) + (l\text{-}0\hat{}2 + 5)\hat{}2 + 12*l\text{-}0\hat{}2 > 0$
using *0* **by** (*simp add*: *algebra-simps power2-eq-square power4-eq-xxxx*)
next
assume $4*x*(x - (7*l\text{-}0\hat{}2 + 5)) + (l\text{-}0\hat{}2 + 5)\hat{}2 + 12*l\text{-}0\hat{}2 > 0$
hence $(2*x - l\text{-}0\hat{}2 - 5)\hat{}2 > (2*l\text{-}0 * sqrt\ (6*x - 3))\hat{}2$
using *0* **by** (*simp add*: *algebra-simps power2-eq-square power4-eq-xxxx*)
have $m > 0$ **using** *assms*(*1*) **by** *simp*
hence $x \geq 7*l\hat{}2*m\hat{}2$
unfolding *x-def* **using** *asm2 assms*(*1*)
by (*simp add*: *mult-imp-le-div-pos power2-eq-square power3-eq-cube*)

hence *4*: *2*x − l-0^2 − 5 ≥ 14*l^2*m^2 − (l*m−1/6)^2 − 5*
by (*simp add*: *x-def l-0-def power2-eq-square*)
have *(l*m−(1/6::real))^2 = (l*m)^2 − l*m/3 + (1/36::real)*
apply (*simp add*: *power2-eq-square*)
by *argo*
hence *14*l^2*m^2 − (l*m−1/6)^2 − 5 = 14*l^2*m^2 − l^2*m^2 + l*m/3 − 1/36 − 5*
using *4* **by** (*auto simp add*: *power2-eq-square*)
hence *14*l^2*m^2 − l^2*m^2 + l*m/3 − 1/36 − 5 = 13*l^2*m^2 + l*m/3 − 1/36 − 5* **by** *argo*
from *asm1 assms(1)* **have** *5*: *l*m/3 > 0* **by** *simp*
have *l > 0* **using** *asm1* **by** *auto*
hence *l*l ≥ 2*2* **using** *asm1 mult-mono' zero-le-numeral* **by** *blast*
have *m > 0* **using** *assms(1)* **by** *auto*
hence *m*m ≥ 3*3*
by (*metis assms(1) less-eq-real-def mult-le-less-imp-less zero-less-numeral*)
hence *13*m*m − 1 ≥ 13*3*3−1* **by** *simp*
have *3*3 > (0::real)* **by** *auto*
hence *13*l*l*m*m ≥(13::real)*2*2*3*3* **using** ‹*l*l ≥ 2*2*› *asm1*
by (*meson* ‹*0 < l*› ‹*0 < m*› *assms(1) less-eq-real-def mult-mono split-mult-pos-le zero-le-numeral*)
hence *13*l^2*m^2 + l*m/3 − 1/36 − 5 ≥ 13*2*2*3*3−1/36−(5::real)*
using *5* **by** (*auto simp add*: *power2-eq-square*)
have *13*3*3*3*3−1/36−(5::real) > 0* **by** *auto*
hence *2*x − l-0^2 − 5 > 0*
using *4* ‹*13 * 2 * 2 * 3 * 3 − 1 / 36 − 5 ≤ 13 * l² * m² + l * m / 3 − 1 / 36 − 5*› ‹*14 * l² * m² − (l * m − 1 / 6)² − 5 = 14 * l² * m² − l² * m² + l * m / 3 − 1 / 36 − 5*› **by** *force*
thus *2*x − l-0^2 − 5 > 2*l-0 * sqrt (6*x − 3)*
by (*smt* (*verit*) ‹*(2 * l-0 * sqrt (6 * x − 3))² < (2 * x − l-0² − 5)²*› *power-mono*)
qed
have *(1/6)^2 * (36::real) = 1* **by** (*auto simp add*: *power2-eq-square*)
from *assms(1)* **have** *m > 0* **by** *auto*
hence *x ≥ 7*l^2*m^2* **unfolding** *x-def* **using** *asm2*
by (*simp add*: *pos-le-divide-eq power2-eq-square power3-eq-cube*)
from *asm1* **have** *l > 0* **by** *auto*
from *assms(1) asm1* ‹*m > 0*› ‹*l > 0*› **have** *l*m ≥ 2*(3::real)*
by (*metis mult-less-cancel-right mult-mono verit-comp-simplify1(1) verit-comp-simplify1(3) zero-le-numeral*)
hence *−2*7*l*m/6 + 7*(1/6)*(1/6) + 5 < (0::real)* **by** *simp*
hence *7*l^2*m^2 > 7*l-0^2 + (5::real)* **unfolding** *l-0-def*
apply (*auto simp add*: *power2-eq-square*)
by *argo*
hence *x ≥ 7*l-0^2 + 5*
using ‹*7 * l² * m² ≤ x*› **by** *linarith*
hence *4*x*(x − (7*l-0^2 + 5)) + (l-0^2 + 5)^2 + 12*l-0^2 > 0*
by (*smt* (*verit*) *mult-nonneg-nonneg power2-less-eq-zero-iff zero-le-power2*)
thus *?thesis*

using *2 3* ‹$(2 * l\text{-}0 * sqrt\ (6 * x - 3) < 2 * x - l\text{-}0^2 - 5) = (0 < 4 * x * (x - (7 * l\text{-}0^2 + 5)) + (l\text{-}0^2 + 5)^2 + 12 * l\text{-}0^2)$› **by** *fastforce*
qed
qed

lemmas *interval-length-greater-than-2m* [*simp*] = *interval-length-greater-than-lm* [**where** *l=2*, *simplified*]

1.2 Lemma 1.11 in [2]

We show Lemma 1.11 in [2] which is also known as Cauchy's Lemma.

theorem *Cauchy-lemma*:
fixes *m N a b r* :: *real*
assumes $m \geq 3$ $N \geq 2*m$
and $0 \leq a$ $0 \leq b$ $0 \leq r$ $r < m$
and $N = m*(a - b)/2 + b + r$
and $1/2 + sqrt\ (6*N/m - 3) \leq b \wedge b \leq 2/3 + sqrt\ (8*N/m - 8)$
shows $b\hat{}2 < 4*a \wedge 3*a < b\hat{}2 + 2*b + 4$

proof−
from *assms* **have** *asm1*: $1/2 + sqrt\ (6*N/m - 3) \leq b$ **and** *asm2*: $b \leq 2/3 + sqrt\ (8*N/m - 8)$ **by** *auto*
have $N - b - r = m*(a - b)/2$ **using** *assms*(*7*) **by** *simp*
hence $a = (N - b - r)*2/m + b$ **using** *assms*(*1*) **by** *simp*
hence $a = b - 2/m * b + 2 * (N - r)/m$
apply (*simp add*: *algebra-simps*)
by (*smt* (*verit*, *del-insts*) *add-divide-distrib*)
hence *a*: $a = b*(1 - 2/m) + 2*(N - r)/m$
by (*simp add*: *right-diff-distrib'*)
have $b\hat{}2 < 4*a$
proof −
from *a* **have** *0*: $b\hat{}2 - 4*a = b\hat{}2 - 4*(1 - 2/m)*b - 8*(N-r)/m$ **by** *simp*
have $3/m \leq 1$ **using** *assms*(*1*) **by** *simp*
hence *1*: $2/3 \leq 2*(1 - 2/m)$ **by** *simp*
have $N/m - 1 < N/m - r/m$ **using** *assms*(*1*,*6*) **by** *simp*
hence $sqrt(8*(N/m - 1)) < sqrt\ (8*((N - r)/m))$ **by** (*simp add*: *diff-divide-distrib*)
hence *2*: $sqrt(8*N/m - 8) < sqrt\ (8*((N - r)/m))$ **by** *simp*
have $2/3 + sqrt\ (8*N/m - 8) < 2*(1 - 2/m) + sqrt\ (8*((N-r)/m))$
using *1 2* **by** *linarith*
hence $b < 2*(1-2/m) + sqrt\ (8*(N - r)/m)$ **using** *asm2* **by** *simp*
hence *3*: $b < 2*(1-2/m) + sqrt\ (4*(1-2/m)\hat{}2 + 8*(N - r)/m)$
by (*smt* (*verit*, *best*) *power2-less-0 real-sqrt-le-iff*)
define *r1* **where** $r1 = 2*(1-2/m) - sqrt\ (4*(1-2/m)\hat{}2 + 8*(N - r)/m)$
define *r2* **where** $r2 = 2*(1-2/m) + sqrt\ (4*(1-2/m)\hat{}2 + 8*(N - r)/m)$

have $r1*r2 = (2*(1-2/m) - sqrt\ (4*(1-2/m)\hat{}2 + 8*(N - r)/m))*(2*(1-2/m) + sqrt\ (4*(1-2/m)\hat{}2 + 8*(N - r)/m))$
using *r1-def r2-def* **by** *simp*
hence $r1*r2 = 2*(1-2/m)*(2*(1-2/m) + sqrt\ (4*(1-2/m)\hat{}2 + 8*(N -$

```
r)/m)) −
sqrt (4*(1−2/m)^2 + 8*(N − r)/m)*(2*(1−2/m) + sqrt (4*(1−2/m)^2 +
8*(N − r)/m))
      by (simp add: Rings.ring-distribs(3))
    hence r1*r2 = (2*(1−2/m))^2+2*(1−2/m)* sqrt (4*(1−2/m)^2 + 8*(N
− r)/m) − 2*(1−2/m)* sqrt (4*(1−2/m)^2 + 8*(N − r)/m)
− (sqrt (4*(1−2/m)^2 + 8*(N − r)/m))^2 by (simp add: distrib-left power2-eq-square)
    hence r1*r2 = (2*(1−2/m))^2 −(sqrt (4*(1−2/m)^2 + 8*(N − r)/m))^2
by simp

    hence r1 * r2 = 4*(1−2/m)^2 − 4*(1−2/m)^2 − 8*(N − r)/m
      using assms(1) assms(2) assms(6) four-x-squared
     by (smt (verit) divide-nonneg-nonneg real-sqrt-pow2-iff zero-compare-simps(12))
    hence r1-times-r2:r1*r2 = −8*(N−r)/m by linarith

     have (b−r1)*(b−r2) = b*(b−r2) − r1*(b−r2) using cross3-simps(28) by
blast
   hence (b−r1)*(b−r2) = b^2−b*r2−b*r1+r1*r2 by (simp add: power2-eq-square
right-diff-distrib)
    hence (b−r1)*(b−r2) = b^2−b*(2*(1−2/m) + sqrt (4*(1−2/m)^2 + 8*(N
− r)/m))−b*(2*(1−2/m) − sqrt (4*(1−2/m)^2 + 8*(N − r)/m))+r1*r2
      using r1-def r2-def by simp
    hence (b−r1)*(b−r2) = b^2−b*2*(1−2/m)−b* sqrt (4*(1−2/m)^2 + 8*(N
− r)/m)−b*(2*(1−2/m) − sqrt (4*(1−2/m)^2 + 8*(N − r)/m)) +r1*r2
      by (simp add: distrib-left)
    hence (b−r1)*(b−r2) = b^2−b*2*(1−2/m)−b* sqrt (4*(1−2/m)^2 + 8*(N
− r)/m)−b*2*(1−2/m)+b* sqrt (4*(1−2/m)^2 + 8*(N − r)/m)+r1*r2
      by (simp add: Rings.ring-distribs(4))
    hence (b−r1)*(b−r2) = b^2−b*4*(1−2/m)+r1*r2 by simp
   hence (b−r1)*(b−r2) = b^2 − 4*(1 − 2/m)*b − 8*(N−r)/m using r1-times-r2
      by (simp add: ‹r1 * r2 = 4 * (1 − 2 / m)^2 − 4 * (1 − 2 / m)^2 − 8 * (N
− r) / m›)
    hence b^2 − 4*(1 − 2/m)*b − 8*(N−r)/m < 0 using 3 assms(4)
     by (smt (verit, del-insts) ‹r1 * r2 = 4 * (1 − 2 / m)^2 − 4 * (1 − 2 / m)^2 −
8 * (N − r) / m› assms(1) assms(2) assms(6) divide-pos-pos mult-nonneg-nonneg
mult-pos-neg r2-def)
    thus ?thesis using 0 by simp
  qed
  have 3*a < b^2 + 2*b + 4
  proof −
    from a have 4: b^2 + 2*b + 4 − 3*a = b^2 − (1−6/m)*b − (6*(N−r)/m
− 4) by argo
    have 5: 1/2 > 1/2 − 3/m using assms(1) by simp
    hence 1/2 − 3/m < 1 by linarith
    also have 1/2 − 3/m > −1 using assms(1)
      by (smt (verit) divide-le-0-1-iff less-divide-eq-1-pos)
    hence (1/2 − 3/m)^2 < 1
     by (metis (no-types, opaque-lifting) calculation less-eq-real-def power2-eq-1-iff
square-le-1 verit-comp-simplify1(3))
```

hence *6*: *sqrt (6∗N/m − 3) > sqrt ((1/2 − 3/m)^2 + 6∗N/m − 4)* **using** *assms(1)* **by** *simp*
from *asm1 5 6* **have** *b > (1/2 − 3/m) + sqrt ((1/2 − 3/m)^2 + 6∗N/m − 4)* **by** *linarith*
hence *7*: *b > (1/2 − 3/m) + sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)*
by (*smt* (*verit*, *ccfv-SIG*) *assms(1) assms(5) divide-right-mono real-sqrt-le-mono*)
define *s1* **where** *s1 = (1/2 − 3/m) − sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)*
define *s2* **where** *s2 = (1/2 − 3/m) + sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)*
have *s1∗ s2=(1/2−3/m)∗((1/2 − 3/m) + sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))−*
sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)∗((1/2 − 3/m) + sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))
using *s1-def s2-def Rings.ring-distribs(3)* **by** *blast*
hence *s1∗ s2= (1/2−3/m)^2+(1/2−3/m)∗ sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)−*
sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)∗((1/2 − 3/m) + sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))
by (*simp add*: *nat-distrib(2) power2-eq-square*)
hence *s1∗ s2= (1/2−3/m)^2+(1/2−3/m)∗ sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)−*
(1/2−3/m)∗ sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4) − (sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))^2
by (*smt* (*verit*, *ccfv-SIG*) *Groups.mult-ac(2) Rings.ring-distribs(3) power2-eq-square*)
hence *8*:*s1∗ s2=(1/2−3/m)^2− (sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))^2* **by** *simp*
from *assms(1,6)* **have** *−r/m > −1* **by** *simp*
hence *−6∗r/m > −6* **by** *simp*
hence *12 − 4 − 6∗r/m > 0* **by** *simp*
hence *12∗m/m − 6∗r/m − 4 > 0* **using** *assms(1)* **by** *simp*
hence *6∗(2∗m − r)/m − 4 > 0* **by** *argo*
hence *6∗(N − r)/m − 4 > 0* **using** *assms(1,2)*
by (*smt* (*verit*, *best*) *divide-right-mono*)
hence *s1 ∗ s2 = (1/2 − 3/m)^2 − (1/2 − 3/m)^2 − 6∗(N − r)/m + 4* **using** *8*
by (*smt* (*verit*) *real-sqrt-pow2-iff zero-le-power2*)

have *(b−s1)∗(b−s2) = b∗(b−s2) − s1∗(b−s2)* **using** *cross3-simps(28)* **by** *blast*
hence *(b−s1)∗(b−s2) = b^2−b∗ s2−b∗ s1+s1∗ s2* **by** (*simp add*: *power2-eq-square right-diff-distrib*)
hence *(b−s1)∗(b−s2) = b^2−b∗((1/2 − 3/m) + sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))−b∗((1/2 − 3/m) − sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))*
+ s1∗ s2 **using** *s1-def s2-def* **by** *simp*
hence *(b−s1)∗(b−s2) = b^2−b∗(1/2 − 3/m)−b∗ sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4)−b∗((1/2 − 3/m) − sqrt ((1/2 − 3/m)^2 + 6∗(N − r)/m − 4))*

+ *s1∗ s2* **by** (*simp add*: *nat-distrib*(*2*))
hence (*b*−*s1*)∗(*b*−*s2*) = *b*^*2*−*b*∗(*1/2* − *3/m*)−*b*∗ *sqrt* ((*1/2* − *3/m*)^*2* + *6*∗(*N* − *r*)/*m* − *4*)−*b*∗(*1/2* − *3/m*)+*b*∗ *sqrt* ((*1/2* − *3/m*)^*2* + *6*∗(*N* − *r*)/*m*−*4*)+*s1*∗ *s2*
by (*smt* (*verit*, *ccfv-SIG*) *nat-distrib*(*2*))
hence (*b*−*s1*)∗(*b*−*s2*) = *b*^*2*−*2*∗*b*∗(*1/2* − *3/m*)+*s1*∗ *s2* **by** *simp*
hence (*b*−*s1*)∗(*b*−*s2*) = *b*^*2*−*2*∗*b*∗(*1/2* − *3/m*)+ (*1/2* − *3/m*)^*2* − (*1/2* − *3/m*)^*2* − *6*∗(*N* − *r*)/*m* + *4*
using ‹*s1* ∗ *s2* = (*1* / *2* − *3* / *m*)^2 − (*1* / *2* − *3* / *m*)^2 − *6* ∗ (*N* − *r*) / *m* + *4*› **by** *fastforce*
hence (*b*−*s1*)∗(*b*−*s2*) = *b*^*2*−*2*∗*b*∗(*1/2*−*3/m*)− *6*∗(*N* − *r*)/*m* + *4* **by** *simp*
hence (*b*−*s1*)∗(*b*−*s2*) = *b*^*2*−*b*∗(*1*−*6/m*)− *6*∗(*N* − *r*)/*m* + *4* **by** *simp*
hence (*b*−*s1*)∗(*b*−*s2*) = *b*^*2*−*b*∗(*1*−*6/m*)− (*6*∗(*N* − *r*)/*m* − *4*) **by** *simp*
hence *b*^*2* − (*1*−*6/m*)∗*b* − (*6*∗(*N*−*r*)/*m* − *4*) > *0* **using** *7* **by** (*smt* (*verit*, *del-insts*) *8 Groups.mult-ac*(*2*)
‹*s1* ∗ *s2* = (*1* / *2* − *3* / *m*)^2 − (*1* / *2* − *3* / *m*)^2 − *6* ∗ (*N* − *r*) / *m* + *4*› *real-sqrt-ge-0-iff s1-def s2-def zero-compare-simps*(*8*) *zero-le-power2*)
thus *?thesis* **using** *4* **by** *simp*
qed
show *?thesis* **by** (*simp add*: ‹*3* ∗ *a* < *b*^2 + *2* ∗ *b* + *4*› ‹*b*^2 < *4* ∗ *a*›)
qed

lemmas *Cauchy-lemma-r-eq-zero* = *Cauchy-lemma* [**where** *r*=*0*, *simplified*]

1.3 Lemma 1.12 in [2]

lemma *not-one*:
fixes *a b* :: *nat*
assumes *a*≥*1*
assumes *b*≥*1*
assumes ∃ *k1* :: *nat*. *a* = *2*∗*k1*+*1*
assumes ∃ *k2* :: *nat*. *b* = *2*∗*k2*+*1*
assumes *b*^*2* < *4*∗*a*
shows *4*∗*a*−*b*^*2* ≠ *1*

proof
assume *4*∗*a*−*b*^*2* = *1*
hence *b*^*2* = *4*∗*a*−*1* **by** *auto*
hence *b*^*2* *mod* *4* = (*4*∗*a*−*1*) *mod* *4* **by** *auto*
have (*4*∗*a*−*1*) *mod* *4* = *3* *mod* *4* **using** *assms*(*1*) **by** (*simp add*: *mod-diff-eq-nat*)
hence *b*^*2* *mod* *4* = *3* **using** ‹*b*^*2* = *4*∗*a*−*1*› *mod-less* **by** *presburger*
thus *False* **using** *assms* **by** (*metis One-nat-def eq-numeral-Suc insert-iff nat.simps*(*3*) *power-two-mod-four pred-numeral-simps*(*3*) *singletonD*)
qed

lemma *not-two*:
fixes *a b* :: *nat*
assumes *a*≥*1*

assumes $b{\geq}1$
assumes $\exists\, k1 :: nat.\ a = 2{*}k1{+}1$
assumes $1{:}\exists\, k2 :: nat.\ b = 2{*}k2{+}1$
assumes $b\hat{}2 < 4{*}a$
shows $4{*}a{-}b\hat{}2 \neq 2$

proof
assume $4{*}a{-}b\hat{}2{=}2$
hence $b\hat{}2{=}4{*}a{-}2$ **by** *auto*
from *1* **have** $2{:}\ \neg\ 2\ dvd\ b\hat{}2$ **by** *auto*
have $2\ dvd\ (4{*}a{-}2)$ **by** *auto*
thus *False* **using** ‹$b\hat{}2{=}4{*}a{-}2$› 2 **by** *auto*
qed

The following lemma shows that given odd positive integers x, y, z and b, where $x \geq y \geq z$, we may pick a suitable integer u where $u = z$ or $u = -z$, such that $b + x + y + u \equiv 0 \pmod 4$.

lemma *suit-z*:
fixes $b\ x\ y\ z :: nat$
assumes $odd\ b \wedge odd\ x \wedge odd\ y \wedge odd\ z$
assumes $x{\geq}y \wedge y{\geq}z$
shows $\exists\ u :: int.\ (u{=}z \vee u{=}{-}z) \wedge (b{+}x{+}y{+}u)\ mod\ 4 = 0$

proof −
from *assms* **have** *0*: $(b{+}x{+}y)\ mod\ 4 = 1 \vee (b{+}x{+}y)\ mod\ 4 = 3$ **by** (*metis dvd-refl even-add*
even-even-mod-4-iff landau-product-preprocess(*53*) *mod-exhaust-less-4*)
from *assms* **have** *1*: $z\ mod\ 4 = 1 \vee z\ mod\ 4 = 3$ **by** (*metis dvd-0-right dvd-refl*
even-even-mod-4-iff mod-exhaust-less-4)

have *c1*:$\exists\, u1{::}int.\ (u1{=}z \vee u1{=}{-}z) \wedge (b{+}x{+}y{+}u1)\ mod\ 4 = 0$
if *asm1*:$(b{+}x{+}y)\ mod\ 4 = 1 \wedge z\ mod\ 4 = 3$
proof −
from *asm1* **have** *2*:$(b{+}x{+}y{+}z)\ mod\ 4 = 0$ **by** (*metis add-num-simps*(*1*) *add-num-simps*(*7*)
mod-add-eq mod-self numeral-plus-one one-plus-numeral-commute)
define *u1* :: *int* **where** *u1*=*z*
show $\exists\, u1{::}int.\ (u1{=}z \vee u1{=}{-}z) \wedge (b{+}x{+}y{+}u1)\ mod\ 4 = 0$ **using** *2 u1-def*
by (*metis Num.of-nat-simps*(*4*) *of-nat-0 of-nat-numeral zmod-int*)
qed

have *c2*:$\exists\, u2{::}int.(u2{=}z \vee u2{=}{-}z) \wedge (b{+}x{+}y{+}u2)\ mod\ 4 = 0$
if *asm2*:$(b{+}x{+}y)\ mod\ 4 = 1 \wedge z\ mod\ 4 = 1$
proof −
from *asm2* **have** *3*:$(b{+}x{+}y{-}z)\ mod\ 4 = 0$
by (*metis assms*(*2*) *mod-eq-0-iff-dvd mod-eq-dvd-iff-nat trans-le-add2*)
define *u2*::*int* **where** *u2*=$-z$
show $\exists\, u2{::}int.(u2{=}z \vee u2{=}{-}z) \wedge (b{+}x{+}y{+}u2)\ mod\ 4 = 0$ **using** *3 u2-def*
by (*metis Num.of-nat-simps*(*2*) *asm2 mod-0*

mod-add-cong more-arith-simps(4) *of-nat-numeral zmod-int*)
qed

have *c3*:∃ *u3*::*int*.(*u3*=*z* ∨ *u3*=−*z*) ∧ (*b*+*x*+*y*+*u3*) *mod* 4 = 0
if *asm3*:(*b*+*x*+*y*) *mod* 4 = 3 ∧ *z mod* 4 = 1
proof −
from *asm3* **have** 4: (*b*+*x*+*y*+*z*) *mod* 4 = 0 **by** (*metis add-num-simps*(1) *add-num-simps*(7)
mod-add-eq mod-self numeral-plus-one)
define *u3*::*int* **where** *u3*=*z*
show ∃ *u3*::*int*.(*u3*=*z* ∨ *u3*=−*z*) ∧ (*b*+*x*+*y*+*u3*) *mod* 4 = 0 **using** 4 *u3-def*
by (*metis Num.of-nat-simps*(4) *of-nat-0 of-nat-numeral zmod-int*)
qed

have *c4*:∃ *u4*::*int*.(*u4*=*z* ∨ *u4*=−*z*) ∧ (*b*+*x*+*y*+*u4*) *mod* 4 = 0
if *asm4*:(*b*+*x*+*y*) *mod* 4 = 3 ∧ *z mod* 4 = 3
proof −
from *asm4* **have** 5: (*b*+*x*+*y*−*z*) *mod* 4 = 0
by (*metis assms*(2) *mod-eq-0-iff-dvd mod-eq-dvd-iff-nat trans-le-add2*)
define *u4*::*int* **where** *u4*=−*z*
show ∃ *u4*::*int*.(*u4*=*z* ∨ *u4*=−*z*) ∧ (*b*+*x*+*y*+*u4*) *mod* 4 = 0 **using** 5 *u4-def*
by (*metis asm4 mod-0*
mod-add-cong more-arith-simps(4) *of-nat-numeral zmod-int*)
qed

show *?thesis* **using** *assms 0 1 c1 c2 c3 c4* **by** *auto*
qed

lemma *four-terms-bin-exp-allsum*:
fixes *b s t u v* :: *int*
assumes *b* = *s*+*t*+*u*+*v*
shows *b*^2 = *t*^2+*u*^2+*s*^2+*v*^2+2∗*t*∗*u*+2 ∗ *s* ∗ *v* + 2∗*t* ∗ *s* + 2∗*t* ∗ *v* +2∗*u* ∗ *s* +2∗*u* ∗ *v*

proof −
from *assms* **have** *b*^2 = (*t*+*u*)^2+(*s*+*v*)^2+2∗(*t*+*u*)∗(*s*+*v*) **by** (*smt* (*verit*, *best*) *power2-sum*)
hence *b-simp1*:*b*^2 = (*t*^2+*u*^2+2∗*t*∗*u*) + (*s*^2+*v*^2+2 ∗ *s* ∗ *v*)+2∗(*t*+*u*)∗(*s*+*v*)

by (*simp add*: *power2-sum*)
have 2∗(*t*+*u*)∗(*s*+*v*) = 2∗*t* ∗ *s* + 2∗*t* ∗ *v* +2∗*u* ∗ *s* +2∗*u* ∗ *v*
using *int-distrib*(1) *int-distrib*(2) **by** *force*
from *this b-simp1* **have** *b-expression*:*b*^2 = *t*^2+*u*^2+*s*^2+*v*^2+2∗*t*∗*u*+2 ∗ *s* ∗ *v* +
2∗*t* ∗ *s* + 2∗*t* ∗ *v* +2∗*u* ∗ *s* +2∗*u* ∗ *v* **by** *auto*
thus *?thesis* **by** *auto*
qed

lemma *four-terms-bin-exp-twodiff*:

fixes *b s t u v* :: *int*
assumes *b = s+t−u−v*
shows *b^2 = t^2+u^2+s^2+v^2−2∗t∗u−2 ∗ s ∗ v + 2∗t ∗ s − 2∗t ∗ v −2∗u ∗ s +2∗u ∗ v*

proof −
from *assms* **have** *b^2 = (s−u)^2+(t−v)^2+2∗(s−u)∗(t−v)* **by** (*smt* (*verit*, *best*) *power2-sum*)
hence *b-simp1*:*b^2 = s^2+u^2−2 ∗ s ∗u + t^2+v^2 − 2 ∗ t ∗ v + 2∗(s−u)∗(t−v)*
by (*simp add*: *power2-diff*)
have *2∗(s−u)∗(t−v) = 2∗ s ∗ t − 2∗ s ∗ v − 2∗u∗t+2∗u ∗ v*
by (*simp add*: *Rings.ring-distribs*(*3*) *Rings.ring-distribs*(*4*))
from *this b-simp1* **have** *b-expression*: *b^2 = t^2+u^2+s^2+v^2−2∗t∗u−2 ∗ s ∗ v +*
2∗t ∗ s − 2∗t ∗ v −2∗u ∗ s +2∗u ∗ v **by** *auto*
thus *?thesis* **by** *auto*
qed

If a quadratic with positive leading coefficient is always non-negative, its discriminant is non-positive.

lemma *qua-disc*:
fixes *a b c* :: *real*
assumes *a>0*
assumes ∀ *x*::*real*. *a∗x^2+b∗x+c ≥0*
shows *b^2 − 4∗a∗c ≤ 0*

proof −
from *assms* **have** *0*:∀ *x*::*real*. *(a∗x^2+b∗x+c)/a ≥0* **by** *simp*
from *assms* **have** *1*:∀ *x*::*real*.*(b∗x+c)/a = b/a∗x+c/a* **by** (*simp add*: *add-divide-distrib*)
from *assms* **have** ∀ *x*::*real*.*(a∗x^2+b∗x+c)/a = x^2+(b∗x+c)/a* **by** (*simp add*: *is-num-normalize*(*1*))
from *1 this* **have** ∀ *x*::*real*.*(a∗x^2+b∗x+c)/a = x^2+b/a∗x+c/a* **by** *simp*
hence *atleastzero*:∀ *x*::*real*. *x^2+b/a∗x+c/a ≥0* **using** *0* **by** *simp*

from *assms* **have** *2*:∀ *x*::*real*. *x^2+b/a∗x+c/a = x^2+2∗b/(2∗a)∗x+c/a+b^2/(4∗a^2)−b^2/(4∗a^2)* **by** *simp*
have *simp1*:∀ *x*::*real*.*(x+b/(2∗a))^2 = x^2+2∗b/(2∗a)∗x+(b/(2∗a))^2* **by** (*simp add*: *power2-sum*)
have *(b/(2∗a))^2 = b^2/(4∗a^2)* **by** (*metis four-x-squared power-divide*)
hence ∀ *x*::*real*. *x^2+b/a∗x+c/a = (x+b/(2∗a))^2+c/a−b^2/(4∗a^2)* **using** *2 simp1* **by** *auto*
hence ∀ *x*::*real*. *(x+b/(2∗a))^2+c/a−b^2/(4∗a^2) ≥0* **using** *atleastzero* **by** *presburger*
hence *3*:∀ *x*::*real*. *b^2/(4∗a^2)−c/a≤(x+b/(2∗a))^2* **by** (*smt* (*verit*, *del-insts*))
have ∃ *x*::*real*. *(x+b/(2∗a))^2=0* **by** (*metis diff-add-cancel power-zero-numeral*)
hence *b^2/(4∗a^2)−c/a≤0* **using** *3* **by** *metis*
hence *4*:*4∗a^2∗(b^2/(4∗a^2)−c/a)≤0* **using** *assms* **by** (*simp add*: *mult-nonneg-nonpos*)
have *5*:*4∗a^2∗b^2/(4∗a^2) = b^2* **using** *assms* **by** *simp*
have *6*:*4∗a^2∗c/a = 4∗a∗c* **using** *assms* **by** (*simp add*: *power2-eq-square*)

show *?thesis* **using** *4 5 6 assms* **by** (*simp add*: *Rings.ring-distribs*(*4*))
qed

The following lemma shows for any point on a 3D sphere with radius a, the sum of its coordinates lies between $\sqrt{3a}$ and $-\sqrt{3a}$.

lemma *three-terms-Cauchy-Schwarz*:
fixes *x y z a* :: *real*
assumes *a* > *0*
assumes *x^2+y^2+z^2* = *a*
shows (*x+y+z*)≥−*sqrt*(*3*∗*a*) ∧ (*x+y+z*)≤*sqrt*(*3*∗*a*)

proof −
have *1*:∀ *t*::*real*. (*t*∗*x+1*)^*2* = *t^2*∗*x^2+1+2*∗*t*∗*x* **by** (*simp add*: *power2-sum power-mult-distrib*)
have *2*:∀ *t*::*real*. (*t*∗*y+1*)^*2* = *t^2*∗*y^2+1+2*∗*t*∗*y* **by** (*simp add*: *power2-sum power-mult-distrib*)
have *3*:∀ *t*::*real*. (*t*∗*z+1*)^*2* = *t^2*∗*z^2+1+2*∗*t*∗*z* **by** (*simp add*: *power2-sum power-mult-distrib*)
from *1 2 3* **have** *4*:∀ *t*::*real*.(*t*∗*x+1*)^*2*+(*t*∗*y+1*)^*2*+(*t*∗*z+1*)^*2* = *t^2*∗*x^2+1+2*∗*t*∗*x* + *t^2*∗*y^2+1+2*∗*t*∗*y* + *t^2*∗*z^2+1+2*∗*t*∗*z* **by** *auto*

have ∀ *t*::*real*. *t^2*∗*x^2*+*t^2*∗*y^2*=*t^2*∗(*x^2*+*y^2*) **by** (*simp add*: *nat-distrib*(*2*))
hence *5*:∀ *t*::*real*. *t^2*∗*x^2*+*t^2*∗*y^2*+*t^2*∗*z^2*=*t^2*∗(*x^2*+*y^2*+*z^2*) **by** (*metis nat-distrib*(*2*))
have *6*:∀ *t*::*real*. *2*∗*t*∗*x*+*2*∗*t*∗*y*+*2*∗*t*∗*z* = *t*∗*2*∗(*x*+*y*+*z*) **by** (*simp add*: *Groups.mult-ac*(*2*) *distrib-right*)
from *4 5 6* **have** ∀ *t*::*real*.(*t*∗*x+1*)^*2*+(*t*∗*y+1*)^*2*+(*t*∗*z+1*)^*2* = *t^2*∗(*x^2*+*y^2*+*z^2*)+ *t*∗*2*∗(*x*+*y*+*z*)+*3*
by (*smt* (*verit*, *best*))
hence ∀ *t*::*real*. *t^2*∗(*x^2*+*y^2*+*z^2*)+ *t*∗*2*∗(*x*+*y*+*z*)+*3* ≥*0* **by** (*metis add-nonneg-nonneg zero-le-power2*)
hence (*2*∗(*x*+*y*+*z*))^*2* − *12*∗(*x^2*+*y^2*+*z^2*)≤*0* **using** *qua-disc*
by (*smt* (*z3*) *power2-diff power2-sum power-zero-numeral sum-squares-bound*)
hence *12*∗(*x^2*+*y^2*+*z^2*)≥*4*∗(*x*+*y*+*z*)^*2* **by** (*simp add*: *four-x-squared*)
hence *3*∗*a*≥(*x*+*y*+*z*)^*2* **using** *assms* **by** *auto*
thus *?thesis* **by** (*smt* (*verit*, *del-insts*) *real-sqrt-abs real-sqrt-le-iff*)
qed

We adapt the lemma above through changing the types for the convenience of our proof.

lemma *three-terms-Cauchy-Schwarz-nat-ver*:
fixes *x y z a* :: *nat*
assumes *a*>*0*
assumes *x^2+y^2+z^2* = *a*
shows (*x+y+z*)≥−*sqrt*(*3*∗*a*) ∧ (*x+y+z*)≤*sqrt*(*3*∗*a*)

proof −
have *fac1*:*real*(*x+y+z*) = *real x* + *real y* + *real z* **by** *auto*

have *fac2*: *3∗(real a) = real(3∗a)* **by** *auto*
thus *?thesis* **using** *fac1 three-terms-Cauchy-Schwarz fac2* **by** (*smt* (*verit*) *assms(1) assms(2) nat-less-real-le of-nat-0-le-iff of-nat-add of-nat-power*)
qed

This theorem is Lemma 1.12 in [2], which shows for odd positive integers a and b satisfying certain properties, there exist four non-negative integers s, t, u and v such that $a = s^2 + t^2 + u^2 + v^2$ and $b = s + t + u + v$. We use the Three Squares Theorem AFP entry [1].

theorem *four-nonneg-int-sum*:
fixes *a b* :: *nat*
assumes *a≥1*
assumes *b≥1*
assumes *odd a*
assumes *odd b*
assumes *3:b^2 < 4∗a*
assumes *3∗a < b^2+2∗b+4*
shows ∃ *s t u v* :: *int. s ≥ 0 ∧ t ≥ 0 ∧ u ≥ 0 ∧ v ≥ 0 ∧ a = s^2 + t^2 + u^2 + v^2 ∧*
b = s+t+u+v

proof −
from *assms* **have** *0:*∃ *k1* :: *nat. a = 2∗k1+1* **by** (*meson oddE*)
from *assms* **have** *1:*∃ *k2* :: *nat. b = 2∗k2+1* **by** (*meson oddE*)
from *0* **have** *4∗a mod 8 = 4* **by** *auto*
hence *2:8 dvd (4∗a−4)* **by** (*metis dvd-minus-mod*)

obtain *k2* **where** *b = 2∗k2+1* **using** *1* **by** *auto*
have *2 dvd k2∗(k2+1)* **by** *auto*
hence *8 dvd 4∗k2∗(k2+1)* **by** (*metis ab-semigroup-mult-class.mult-ac(1) mult-2-right nat-mult-dvd-cancel-disj numeral-Bit0*)
hence *b^2 mod 8 = 1* **using** *1* **by** (*metis One-nat-def Suc-0-mod-numeral(2) assms(4) square-mod-8-eq-1-iff unique-euclidean-semiring-class.cong-def*)
hence *8 dvd (b^2−1)* **by** (*metis dvd-minus-mod*)
from *2 this* **have** *8 dvd ((4∗a−4)−(b^2−1))* **using** *dvd-diff-nat* **by** *blast*
from *assms 0 1* **and** *this* **have** *7:8 dvd ((4∗a−b^2)−3)* **by** *auto*
from *assms 0 1* **have** *5:4∗a−b^2≠1* **using** *not-one* **by** *auto*
from *assms 0 1* **have** *6:4∗a−b^2≠2* **using** *not-two* **by** *auto*
from *3 5 6* **have** *4∗a−b^2 ≥ 3* **by** *auto*
from *this 7* **have** *8:(4∗a−b^2) mod 8 = 3* **using** *mod-nat-eqI* **by** *presburger*

obtain *j k l* **where** *ints:odd j ∧ odd k ∧ odd l ∧ (4∗a−b^2) = j^2+k^2+l^2*
using *8 odd-three-squares-using-mod-eight* **by** *presburger*
define *x* **where** *x = sort[j,k,l] ! 2*
define *y* **where** *y = sort[j,k,l] ! 1*
define *z* **where** *z = sort[j,k,l] ! 0*

have *x^2+y^2+z^2 = sum-list (map (λx. x^2) [j,k,l])* **using** *x-def y-def z-def*

by *auto*
from *this ints* **have** *a-and-b*:$(4*a-b\hat{}2) = x\hat{}2+y\hat{}2+z\hat{}2$ **by** *auto*

have *size*:$x \geq y \wedge y \geq z$ **using** *x-def y-def z-def* **by** *auto*
have *x-par*:$x = j \vee x = k \vee x = l$ **using** *x-def* **by** *auto*
have *y-par*:$y = j \vee y = k \vee y = l$ **using** *y-def* **by** *auto*
have *z-par*:$z = j \vee z = k \vee z = l$ **using** *z-def* **by** *auto*
hence *parity*:*odd x* $\wedge$ *odd y* $\wedge$ *odd z* **using** *ints x-par y-par z-par* **by** *fastforce*
from *1* **have** *b-par*:*odd b* **by** *auto*

obtain *w*::*int* **where** *w-def*:$(w=z \vee w=-z) \wedge (b+x+y+w) \bmod 4 = 0$
using *suit-z size parity b-par* **by** *presburger*

from *parity* **have** *fac1*:$(int\ z) \bmod 4 = 3 \vee (int\ z) \bmod 4 = 1$ **by** *presburger*
from *parity* **have** *fac2*:$-z \bmod 4 = 3 \vee -z \bmod 4 = 1$ **by** *presburger*
from *w-def* **have** *fac3*:$w \bmod 4 = 3 \vee w \bmod 4 = 1$ **using** *fac1 fac2* **by** *auto*

have *s-int*:$4\ dvd\ (b+x+y+w)$ **using** *b-par parity fac3 w-def* **by** *presburger*
have *b-x-int*:$2\ dvd\ (b+x)$ **using** *b-par parity* **by** *presburger*
have *b-y-int*:$2\ dvd\ (b+y)$ **using** *b-par parity* **by** *presburger*
have *b-w-int*:$2\ dvd\ (b+w)$ **using** *b-par fac3* **by** *presburger*

obtain *s*::*int* **where** *s-def*:$s = (b+x+y+w)\ div\ 4$ **using** *s-int* **by** *fastforce*
obtain *t*::*int* **where** *t-def*:$t = (b+x)\ div\ 2 - s$ **using** *s-int b-x-int* **by** *blast*
obtain *u*::*int* **where** *u-def*:$u = (b+y)\ div\ 2 - s$ **using** *s-int b-y-int* **by** *blast*
obtain *v*::*int* **where** *v-def*:$v = (b+w)\ div\ 2 - s$ **using** *s-int b-w-int* **by** *blast*

from *t-def s-def* **have** *t-simp1*:$t = (2*b+2*x)\ div\ 4 - (b+x+y+w)\ div\ 4$ **by** *auto*
have *t-simp2*:$(2*\ b+2*\ x) - (b+x+y+w) = b+x-y-w$ **using** *size* **by** *auto*
hence *t-expre*:$t = (b+x-y-w)\ div\ 4$ **using** *t-simp1* **by** (*smt* (*verit*, *ccfv-SIG*) *add-num-simps*(*1*) *div-plus-div-distrib-dvd-right numeral-Bit0 of-nat-numeral one-plus-numeral s-int unique-euclidean-semiring-with-nat-class.of-nat-div*)
from *b-x-int* **have** *4 dvd* $(2*b+2*x)$
by (*metis distrib-left-numeral mult-2-right nat-mult-dvd-cancel-disj numeral-Bit0*)
hence *four-div-tn*:$4\ dvd\ (b+x-y-w)$ **using** *s-int t-simp2* **by** *presburger*

have $(b+x)\ div\ 2 + (b+y)\ div\ 2 = (2*b+x+y)\ div\ 2$
by (*smt* (*verit*, *best*) *Groups.add-ac*(*2*) *b-y-int div-plus-div-distrib-dvd-right left-add-twice nat-arith.add2*)
hence *threesum*:$t + u + s = (2*b+x+y)\ div\ 2 - s$ **using** *t-def u-def* **by** *auto*

have *2 dvd* $(x+y)$ **using** *parity* **by** *auto*
hence $(2*b+x+y)\ div\ 2 + (b+w)\ div\ 2 = (2*b+b+x+y+w)\ div\ 2$
by (*smt* (*verit*, *ccfv-threshold*) *Num.of-nat-simps*(*4*) *b-w-int div-plus-div-distrib-dvd-right landau-product-preprocess*(*4*) *numerals*(*1*) *of-nat-1 one-plus-numeral unique-euclidean-semiring-with-nat-class.of-nat-div*)

hence $t+u+s+v = (2*b+b+x+y+w)$ *div* $2 - s - s$ **using** *v-def threesum* **by** *auto*
hence *foursum0*:$t+u+s+v = (2*b+b+x+y+w)$ *div* $2 - (b+x+y+w)$ *div* $4 - (b+x+y+w)$ *div* 4
using *s-def* **by** *auto*
have *foursum1*:$(b+x+y+w)$ *div* $4 + (b+x+y+w)$ *div* $4 = (b+x+y+w)$ *div* 2
using *div-mult-swap s-int* **by** *auto*
have $(2*b+b+x+y+w)$ *div* $2 - (b+x+y+w)$ *div* $2 = (2*b)$ *div* 2 **by** *auto*
hence $t+u+s+v = (2*b)$ *div* 2 **using** *foursum0 foursum1* **by** *linarith*
hence *second*:$t+u+s+v = b$ **by** *auto*

from *a-and-b* **have** $4*a = x^2+y^2+z^2+b^2$
by (*metis Nat.add-diff-assoc2 add-diff-cancel-right' assms*(5) *less-or-eq-imp-le*)
hence $a = (x^2+y^2+z^2+b^2)$ *div* 4 **using** *parity b-par* **by** *auto*

from *second* **have** *b-expresion*:$b^2 = t^2+u^2+s^2+v^2+2*t*u+2* s * v + 2*t* s + 2*t * v +2*u * s +2*u * v$ **using** *four-terms-bin-exp-allsum*
by (*metis is-num-normalize*(1) *nat-arith.add2 of-nat-power*)

define *sn* **where** *sn-def*:$sn = b+x+y+w$
from *sn-def s-def* **have** *sn-nume*: $4* s = sn$ **by** (*metis dvd-div-mult-self mult.commute s-int*)
from *sn-def* **have** *sn-sqr*:$sn^2 = b^2+x^2+y^2+w^2+2* b * x+2* b * y+2*b*w+2*x*y+2*x*w+2*y*w$

using *four-terms-bin-exp-allsum w-def* **by** *auto*
hence *s-pen*:$16* s^2 = b^2+x^2+y^2+w^2+2*b*x+2*b*y+2*b*w+2*x*y+2*x*w+2*y*w$ **using** *sn-nume* **by** *auto*
have *4 dvd sn* **using** *s-int sn-def* **by** *auto*
hence *16 dvd sn^2* **by** *auto*
hence *s-sqr-expression*:$s^2=(b^2+x^2+y^2+w^2+2*b*x+2*b*y+2*b*w+2*x*y+2*x*w+2*y*w)$ *div 16*
using *sn-sqr s-pen* **by** *auto*

define *tn* **where** *tn-def*:$tn = b+x-y-w$
from *tn-def t-expre size four-div-tn* **have** *tn-nume*: $4* t = tn$
by (*metis dvd-div-mult-self mult.commute*)
from *size assms* **have** $b+x-y > 0$ **by** *auto*
hence $tn = int\ b + int\ x - int\ y - w$ **using** *tn-def* **by** *auto*
from *this* **have** *tn-sqr*:$tn^2 = b^2+x^2+y^2+w^2+2*b*x-2*b*y-2*b*w-2*x*y-2*x*w+2*y*w$
using *four-terms-bin-exp-twodiff w-def* **by** *auto*
hence *t-pen*:$16*t^2 = b^2+x^2+y^2+w^2+2*b*x-2*b*y-2*b*w-2*x*y-2*x*w+2*y*w$ **using** *tn-nume* **by** *auto*
have *16 dvd tn^2* **using** *tn-def four-div-tn* **by** *auto*
hence *t-sqr-expression*:$t^2=(b^2+x^2+y^2+w^2+2*b*x-2*b*y-2*b*w-2*x*y-2*x*w+2*y*w)$ *div 16*
using *tn-sqr t-pen* **by** *auto*

from *size s-def t-expre w-def* **have** *sgeqt*:$s \geq t$ **by** *auto*
from *size s-def t-def u-def* **have** *tgequ*:$t \geq u$ **by** *auto*
from *size s-def u-def v-def w-def* **have** *ugeqv*:$u \geq v$ **by** *auto*

from *assms*(*6*) **have** *12*∗*a* < *4*∗*b*^*2*+*8*∗*b*+*16* **by** *auto*
hence *12*∗*a*−*3*∗*b*^*2* < *b*^*2*+*8*∗*b*+*16* **by** *auto*
hence *12*∗*a*−*3*∗*b*^*2* < (*b*+*4*)^*2*
by (*smt* (*z3*) *add.commute add.left-commute mult-2 numeral-Bit0 power2-eq-square power2-sum*)
hence *mid-ineq*:*sqrt*(*12*∗*a*−*3*∗*b*^*2*) < *b*+*4*
by (*meson of-nat-0-le-iff of-nat-power-less-of-nat-cancel-iff real-less-lsqrt*)

define *ab*::*nat* **where** *ab-def*:*ab* = *4*∗*a*−*b*^*2*
from *assms ab-def* **have** *nonneg-ab*:*ab*>*0* **by** *auto*
from *a-and-b ab-def* **have** *sum-of-sqrs*:*x*^*2*+*y*^*2*+*z*^*2* = *ab* **by** *auto*
from *this nonneg-ab* **have** *x*^*2*+*y*^*2*+*z*^*2*>*0* **by** *auto*
from *this sum-of-sqrs three-terms-Cauchy-Schwarz-nat-ver* **have** *x*+*y*+*z* ≤ *sqrt*(*3*∗*ab*) **by** *auto*
hence *left-ineq*:*x*+*y*+*z* ≤ *sqrt*(*3*∗(*4*∗*a*−*b*^*2*)) **using** *ab-def* **by** *auto*
have *sqrt*(*3*∗(*4*∗*a*−*b*^*2*)) = *sqrt*(*12*∗*a*−*3*∗*b*^*2*) **by** (*simp add*: *diff-mult-distrib2*)
from *left-ineq mid-ineq this* **have** *x*+*y*+*z* < *b*+*4* **by** *auto*
hence *num-bound*:*int b*− *x*− *y*− *z* > −*4* **by** *auto*

define *vn* **where** *vn-def*:*vn* = *int b*+*w*−*x*−*y*
from *num-bound vn-def w-def* **have** *vn-bound*:*vn* > −*4* **by** *auto*
from *w-def* **have** *four-div-sn*:*4 dvd* (*int b* +*x*+*y*+*w*) **by** *auto*
from *parity* **have** *4 dvd* (*int 2*∗*x*+*2*∗*y*)
by (*metis Num.of-nat-simps*(*5*) ‹*even* (*x* + *y*)› *distrib-left int-dvd-int-iff nat-mult-dvd-cancel-disj num-double numeral-mult of-nat-add of-nat-numeral*)
hence *4 dvd* (*int b* +*x*+*y*+*w* − *2*∗*x* − *2*∗*y*) **using** *four-div-sn*
by (*smt* (*verit*) *Num.of-nat-simps*(*5*) *dvd-add-left-iff*)
hence *4 dvd vn* **using** *vn-def* **by** *presburger*

from *v-def s-def* **have** *v* = (*int 2*∗*b* + *2*∗*w*) *div 4* − (*int b* + *x* + *y* + *w*) *div 4* **by** *auto*
hence *v-expre*:*v* = (*int b*−*x*−*y*+*w*) *div 4* **using** *four-div-sn* **by** *fastforce*
hence *v* = *vn div 4* **using** *vn-def* **by** *auto*
hence *v* ≥ *0* **using** *vn-bound four-div-sn* **using** ‹*4 dvd vn*› **by** *fastforce*
hence *stuv-nonneg*: *s* ≥ *0* ∧ *t* ≥ *0* ∧ *u* ≥ *0* ∧ *v* ≥ *0* **using** *sgeqt tgequ ugeqv* **by** *linarith*

from *vn-def* **have** *vn-sqr*:*vn*^*2* = *b*^*2*+*x*^*2*+*y*^*2*+*w*^*2* − *2*∗*b*∗*x*−*2*∗*b*∗*y*+*2*∗*b*∗*w*+*2*∗*x*∗*y*−*2*∗*x*∗*w*−*2*∗*y*∗*w*

using *four-terms-bin-exp-twodiff w-def* **by** *auto*
from ‹*v* = *vn div 4*› **have** *vn-is-num*:*v*^*2* = *vn*^*2 div 16* **using** ‹*4 dvd vn*› **by** *fastforce*
hence *16 dvd vn*^*2* **using** *v-def* **using** ‹*4 dvd vn*› **by** *fastforce*
from *vn-is-num vn-sqr* **have** *v-sqr-expression*:*v*^*2*=(*b*^*2*+*x*^*2*+*y*^*2*+*w*^*2*−*2*∗*b*∗*x*−*2*∗*b*∗*y*+*2*∗*b*∗*w*+*2*∗*x*∗*y*−*2*∗*x*∗*w*−*2*∗*y*∗*w*) *div 16* **by** *auto*

define *un* **where** *un-def*:*un* = *int b*+*y*−*x*−*w*

```
  from parity w-def have even (x+w) by auto
  from this parity have 4 dvd (int 2*x+2*w)
    by (metis distrib-left even-numeral mult-2-right mult-dvd-mono numeral-Bit0
of-nat-numeral)
  hence 4 dvd (int b +x+y+w - 2*x-2*w) using four-div-sn
    by (smt (verit) Num.of-nat-simps(5) dvd-add-left-iff)
  hence 4 dvd un using un-def by presburger

  from u-def s-def have u = (int 2*b+2*y) div 4 - (int b + x + y + w) div 4
by auto
  hence u-expre:u = (int b-x+y-w) div 4 using four-div-sn by fastforce
  hence u = un div 4 using un-def by auto
 from un-def have un-sqr:un^2 = b^2+x^2+y^2+w^2+2*b*y-2*b*x-2*b*w-2*y*x-2*y*w+2*x*w
    using four-terms-bin-exp-twodiff w-def by auto
  from ‹u = un div 4› have un-is-num:u^2 = un^2 div 16 using ‹4 dvd un› by
fastforce
  hence 16 dvd un^2 using u-def using ‹4 dvd un› by fastforce
  from un-is-num un-sqr have
 u-sqr-expression:u^2 = (b^2+x^2+y^2+w^2+2*b*y-2*b*x-2*b*w-2*y*x-2*y*w+2*x*w)
div 16 by auto

  from u-sqr-expression v-sqr-expression have
 uv-simp1:u^2+v^2 = (int b^2+x^2+y^2+w^2-2*b*x-2*b*y+2*b*w+2*x*y-2*x*w-2*y*w)
div 16 +
   (int b^2+x^2+y^2+w^2+2*b*y-2*b*x-2*b*w-2*y*x-2*y*w+2*x*w) div
16 by auto
 have uv-simp2:(int b^2+x^2+y^2+w^2-2*b*x-2*b*y+2*b*w+2*x*y-2*x*w-2*y*w)+
 (int b^2+x^2+y^2+w^2+2*b*y-2*b*x-2*b*w-2*y*x-2*y*w+2*x*w)=
 (int 2*b^2+2*x^2+2*y^2+2*w^2-4*b*x-4*y*w) by auto
 hence 16 dvd (int 2*b^2+2*x^2+2*y^2+2*w^2-4*b*x-4*y*w) by (smt (verit)
‹16 dvd un^2› ‹16 dvd vn^2›
 dvd-add-right-iff of-nat-power un-sqr vn-sqr zadd-int-left)
 hence usqr-plus-vsqr:u^2+v^2 = (int 2*b^2+2*x^2+2*y^2+2*w^2-4*b*x-4*y*w)
div 16
    using uv-simp1 uv-simp2 by (smt (verit, ccfv-threshold) Num.of-nat-simps(4)
Num.of-nat-simps(5)
 ‹16 dvd vn^2› div-plus-div-distrib-dvd-right power2-eq-square vn-sqr)

  have allsum0:s^2+t^2+u^2+v^2 = (sn^2+tn^2+un^2+vn^2) div 16 using
‹16 dvd vn^2› ‹16 dvd sn^2›
‹16 dvd un^2› ‹16 dvd tn^2› s-sqr-expression t-sqr-expression u-sqr-expression v-sqr-expression
sn-sqr tn-sqr un-sqr vn-sqr by (metis add.commute div-plus-div-distrib-dvd-left)
 have allsum1:(sn^2+tn^2+un^2+vn^2) = (int 4*b^2+4*x^2+4*y^2+4*w^2)

    using sn-sqr tn-sqr un-sqr vn-sqr by auto
  have 16 dvd (sn^2+tn^2+un^2+vn^2)
    by (simp add: ‹16 dvd sn^2› ‹16 dvd tn^2› ‹16 dvd un^2› ‹16 dvd vn^2›)
  hence 16 dvd 4*(int b^2+x^2+y^2+w^2) using allsum1 by auto
  hence 4 dvd (int b^2+x^2+y^2+w^2) by presburger
```

from *allsum1* **have** *s^2+t^2+u^2+v^2 = (int 4*b^2+4*x^2+4*y^2+4*w^2) div 16*
using *allsum0* **by** *presburger*
hence *s^2+t^2+u^2+v^2 = 4*(int b^2+x^2+y^2+w^2) div 16* **by** *simp*
hence *allsum2:s^2+t^2+u^2+v^2 = (int b^2+x^2+y^2+w^2) div 4* **by** *simp*

from *a-and-b* **have** *4*a = int b^2+x^2+y^2+w^2* **using** *w-def*
using ‹$4 * a = x^2 + y^2 + z^2 + b^2$› **by** *fastforce*
hence *first:a = s^2+t^2+u^2+v^2* **using** *allsum2* **by** *linarith*

show *?thesis* **using** *first second stuv-nonneg* **by** (*smt* (*verit*, *best*))
qed
end

2 Polygonal Number Theorem

2.1 Gauss's Theorem on Triangular Numbers

We show Gauss's theorem which states that every non-negative integer is the sum of three triangles, using the Three Squares Theorem AFP entry [1]. This corresponds to Theorem 1.8 in [2].

theory *Polygonal-Number-Theorem-Gauss*
imports *Polygonal-Number-Theorem-Lemmas*
begin

The following is the formula for the k-th polygonal number of order $m + 2$.

definition *polygonal-number* :: *nat* ⇒ *nat* ⇒ *nat*
where *polygonal-number m k = m*k*(k−1) div 2 + k*

When $m = 1$, the polygonal numbers have order 3 and the formula represents triangular numbers. Gauss showed that all natural numbers can be written as the sum of three triangular numbers. In other words, the triangular numbers form an additive basis of order 3 of the natural numbers.

theorem *Gauss-Sum-of-Three-Triangles*:
fixes *n* :: *nat*
shows ∃ *x y z*. *n = polygonal-number 1 x + polygonal-number 1 y + polygonal-number 1 z*

proof −
have *(8 * n + 3) mod 8 = 3* **by** *auto*
then obtain *a b c* **where** *0*: *odd a ∧ odd b ∧ odd c ∧ 8 * n + 3 = a^2 + b^2 + c^2*
using *odd-three-squares-using-mod-eight* **by** *presburger*
then obtain *x y z* **where** *a = 2 * x + 1 ∧ b = 2 * y + 1 ∧ c = 2 * z + 1* **by** (*meson oddE*)
hence *8 * n + 3 = (2 * x + 1)^2 + (2 * y + 1)^2 + (2 * z + 1)^2*
using *0* **by** *auto*

hence $n = (x * x + x + y * y + y + z * z + z)$ *div 2*
by (*auto simp add*: *power2-eq-square*)
hence *n-expr*:$n = (x * (x + 1) + y * (y + 1) + z * (z + 1))$ *div 2*
by (*metis* (*no-types*, *lifting*) *arithmetic-simps*(*79*) *nat-arith.add1 nat-distrib*(*2*))

have *triangle-identity*: *polygonal-number 1 k* = *k*∗(*k*+*1*) *div 2* **for** *k*
proof −
have *k*∗(*k*−*1*)+*2*∗*k* = *k*∗*k*+*k* **by** (*simp add*: *right-diff-distrib'*)
hence *k*∗(*k*−*1*) *div 2* + *k* = (*k*∗*k*+*k*) *div 2*
by (*metis Groups.add-ac*(*2*) *bot-nat-0.not-eq-extremum div-mult-self2 pos2*)
thus *?thesis* **using** *polygonal-number-def* **by** *simp*
qed
from *n-expr triangle-identity* **show** *?thesis*
by (*metis div-plus-div-distrib-dvd-right even-mult-iff odd-even-add odd-one*)
qed
end

2.2 Cauchy's Polygonal Number Theorem

We will use the definition of the polygonal numbers from the Gauss Theorem theory file which also imports the Three Squares Theorem AFP entry [1].

theory *Polygonal-Number-Theorem-Cauchy*
imports *Polygonal-Number-Theorem-Gauss*
begin

The following lemma shows there are two consecutive odd integers in any four consecutive integers.

lemma *two-consec-odd*:
fixes *a1 a2 a3 a4* :: *int*
assumes *a1*−*a2* = *1*
assumes *a2*−*a3* = *1*
assumes *a3*−*a4* = *1*
shows ∃ *k1 k2* :: *int*. {*k1*, *k2*} ⊆ {*a1*, *a2*, *a3*, *a4*} ∧ (*k2* = *k1*+*2*) ∧ *odd k1*

proof −
have *c1*:∃ *k1 k2* :: *int*. {*k1*, *k2*} ⊆ {*a1*, *a2*, *a3*, *a4*} ∧ (*k2* = *k1*+*2*) ∧ *odd k1*
if *odd-case*:*odd a4*
proof−
define *k1* **where** *k1-def*:*k1* = *a4*
define *k2* **where** *k2-def*:*k2* = *k1* + *2*
have *0*:*k2* = *a2* **using** *k2-def k1-def assms* **by** *simp*
have *1*:*odd k1* **using** *k1-def odd-case* **by** *simp*
show ∃ *k1 k2* :: *int*. {*k1*, *k2*} ⊆ {*a1*, *a2*, *a3*, *a4*} ∧ (*k2* = *k1*+*2*) ∧ *odd k1*
using *0 1 k1-def k2-def* **by** *auto*
qed

have *c2*:∃ *k1 k2* :: *int*. {*k1*, *k2*} ⊆ {*a1*, *a2*, *a3*, *a4*} ∧ (*k2* = *k1*+*2*) ∧ *odd k1*
if *even-case*:*even a4*

proof −
define *k1* **where** *k1-def*:*k1* = *a3*
define *k2* **where** *k2-def*:*k2* = *k1* + *2*
have *2*:*odd k1* **using** *even-case assms k1-def* **by** *presburger*
have *3*:*k2* = *a1* **using** *k1-def k2-def assms* **by** *simp*
show ∃ *k1 k2* :: *int*. {*k1*, *k2*} ⊆ {*a1*, *a2*, *a3*, *a4*} ∧ (*k2* = *k1*+*2*) ∧ *odd k1*
using *2 3 k1-def k2-def* **by** *auto*
qed
show *?thesis* **using** *c1 c2* **by** *auto*
qed

This lemma proves that for two consecutive integers b_1 and b_2, and $r \in \{0, 1, \ldots, m-3\}$, numbers of the form $b_1 + r$ and $b_2 + r$ can cover all the congruence classes modulo m.

lemma *cong-classes*:
fixes *b1 b2* :: *int*
fixes *m* :: *nat*
assumes *m* ≥ *4*
assumes *odd b1*
assumes *b2* = *b1* + *2*
shows ∀ *N*::*nat*. ∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m*−*3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1* ∨ *b* = *b2*)

proof −
have *first*:∀ *N*::*nat*. ∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m*−*3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1* ∨ *b* = *b2*)
if *first-assum*:*b1 mod m* ≥ *3*
proof −
define *k1* **where** *k1-def*:*k1* = *b1 mod m*
define *l* **where** *l-def*:*l* = *m* − *k1*
have *k1-size*:*k1*≥*3* **using** *first-assum k1-def* **by** *simp*
have *l-size*:*l* ≤ *m*−*3* **using** *first-assum k1-def l-def* **by** *auto*
have (*l*+*k1*) *mod m* = *0* **using** *l-def* **by** *auto*
hence (*l*+*b1*) *mod m* = *0* **using** *k1-def l-def* **by** (*metis mod-add-right-eq*)
define *w* **where** *w-def*:*w* = *m*−*3*−*l*
have *w-size*:*w*≥*0* ∧ *w*≤*m*−*3* **using** *w-def l-size l-def k1-def first-assum*
by (*smt* (*verit*, *best*) *Euclidean-Rings.pos-mod-bound assms*(*1*) *le-antisym numeral-neq-zero*
of-nat-0-less-iff order-trans-rules(*22*) *verit-comp-simplify*(*3*) *zero-le-numeral*)
have *k1* = *w*+*3* **using** *w-def k1-def l-def w-size first-assum* **by** *linarith*
hence *w*+*2* = *k1*−*1* **by** *auto*
hence *w*+*2* = (*b1*−*1*) *mod m* **using** *first-assum k1-def*
by (*smt* (*verit*, *del-insts*) *Euclidean-Rings.pos-mod-bound assms*(*1*)
mod-diff-eq mod-pos-pos-trivial of-nat-le-0-iff verit-comp-simplify(*8*))
hence *w-cover*:*w*+*2* = *k1*−*1* **using** *k1-def* **using** ‹*w* + *2* = *k1* − *1*› **by** *fastforce*

have ∃ *r*::*nat*. (*r*≤*m*−*3*) ∧ [*N*=*b1*+*r*] (*mod m*) **if** *asm1*:*N mod m* ≥ *k1* ∧ *N mod m* ≤ *m*−*1* **for** *N*

proof −
have $m - (N \ mod \ m) \leq l$ **using** *asm1 l-def k1-def* **by** *linarith*
hence $\exists\, d{::}nat.\ d{\leq}l \wedge [N = k1{+}d]\ (mod\ m)$ **using** *asm1 k1-def l-def*
by (*metis add.commute add-le-cancel-left cong-mod-left cong-refl diff-add-cancel diff-le-self le-trans of-nat-mod of-nat-mono zle-iff-zadd*)
hence $\exists\, d{::}nat.\ d{\leq}l \wedge [N{=}b1{+}d]\ (mod\ m)$ **using** *k1-def*
by (*metis mod-add-left-eq unique-euclidean-semiring-class.cong-def*)
thus $\exists\, r{::}nat.\ (r{\leq}m{-}3) \wedge [N{=}b1{+}r]\ (mod\ m)$ **using** *l-size*
by (*smt* (*verit*, *best*) *nat-leq-as-int*)
qed
hence *c1*:$\exists\, b{::}int.\ \exists\, r{::}nat.\ (r \leq m{-}3) \wedge [N{=}b{+}r]\ (mod\ m) \wedge (b = b1 \vee b = b2)$ **if** *asm1*:$N \ mod \ m \geq k1 \wedge N \ mod \ m \leq m{-}1$ **for** N **using** *asm1* **by** *blast*

have *c2*:$\exists\, r{::}nat.\ (r{\leq}m{-}3) \wedge [N{=}b1{+}r]\ (mod\ m)$ **if** *asm2*:$N \ mod \ m = 0$ **for** N **using** *l-def k1-def*
by (*smt* (*verit*, *ccfv-threshold*) ‹$(l + b1) \ mod \ int \ m = 0$› *add-diff-cancel-left' cong-0-iff cong-sym cong-trans diff-add-cancel diff-ge-0-iff-ge dvd-eq-mod-eq-0 int-dvd-int-iff nat-0-le of-nat-le-iff that w-def w-size*)
hence *c2*:$\exists\, b{::}int.\ \exists\, r{::}nat.\ (r \leq m{-}3) \wedge [N{=}b{+}r]\ (mod\ m) \wedge (b = b1 \vee b = b2)$
if *asm2*:$N \ mod \ m = 0$ **for** N **using** *asm2* **by** *metis*

have $\exists\, r{::}nat.\ (r{\leq}m{-}3) \wedge [N{=}b1{+}r]\ (mod\ m)$ **if** *asm3*:$N \ mod \ m > 0 \wedge N \ mod \ m \leq w$ **for** N
proof −
have $l{+}\ (N \ mod \ m) \leq m{-}3$ **using** *asm3 w-def* **by** *auto*
hence $\exists\, d{::}nat.\ (d{\leq}w) \wedge [N = k1{+}l{+}d]\ (mod\ m)$ **using** *asm3 w-def k1-def l-def*
by (*smt* (*verit*, *ccfv-threshold*) *minus-mod-self2 mod-mod-trivial of-nat-mod unique-euclidean-semiring-class.cong-def*)
hence $\exists\, d{::}nat.\ (d{\leq}w) \wedge [N = b1{+}l{+}d]\ (mod\ m)$ **using** *k1-def* **by** (*metis* (*mono-tags*, *opaque-lifting*) *mod-add-left-eq unique-euclidean-semiring-class.cong-def*)
hence $\exists\, r{::}nat.\ (r{\leq}w{+}l) \wedge [N = b1{+}r]\ (mod\ m)$ **by** (*smt* (*verit*) *add.commute add.left-commute le-add-same-cancel2 of-nat-0-le-iff w-def w-size zero-le-imp-eq-int*)
thus $\exists\, r{::}nat.\ (r{\leq}m{-}3) \wedge [N = b1{+}r]\ (mod\ m)$ **using** *w-def* **by** *auto*
qed
hence $\exists\, b{::}int.\ \exists\, r{::}nat.\ (r \leq m{-}3) \wedge [N{=}b{+}r]\ (mod\ m) \wedge (b = b1 \vee b = b2)$
if *asm3*:$N \ mod \ m > 0 \wedge N \ mod \ m \leq w$ **for** N **using** *asm3* **by** *blast*
hence *c3*:$\exists\, b{::}int.\ \exists\, r{::}nat.\ (r \leq m{-}3) \wedge [N{=}b{+}r]\ (mod\ m) \wedge (b = b1 \vee b = b2)$
if *asm8*:$N \ mod \ m > 0 \wedge N \ mod \ m \leq k1{-}3$ **using** *asm8 w-cover* **by** *auto*

have $\exists\, r{::}nat.\ (r{\leq}m{-}3) \wedge [N{=}b2{+}r]\ (mod\ m)$ **if** *asm4*:$N \ mod \ m = w{+}1 \vee N \ mod \ m = w{+}2$ **for** N
proof −
have *c4-1*:$[N = b2{+}(m{-}3)]\ (mod\ m)$ **if** *asm5*:$N \ mod \ m{=}w{+}2$ **for** N **using**

asm5 w-def assms(3) l-def
by (*smt* (*verit*) ‹*w + 2 = (b1 − 1) mod int m*› ‹*w + 2 = k1 − 1*› *mod-add-self1 of-nat-mod*
unique-euclidean-semiring-class.cong-def)
hence [*N−1 = b2+(m−4)*] (*mod m*) **if** *asm5*:*N mod m = w+2* **for** *N*
by (*smt* (*verit*, *ccfv-threshold*) *Num.of-nat-simps*(*2*) ‹*w + 2 = k1 − 1*› *asm5 assms*(*1*)
cong-iff-lin first-assum k1-def l-def mod-less-eq-dividend numeral-Bit0 of-nat-diff of-nat-le-iff
of-nat-numeral semiring-norm(*172*) *w-def*)
hence [*N = b2+(m−4)*] (*mod m*) **if** *asm6*:*N mod m = w+1* **for** *N* **using** *asm6*
by (*metis* ‹*w + 2 = (b1 − 1) mod int m*› *add-diff-cancel-right' arith-special*(*3*) *int-ops*(*4*)
is-num-normalize(*1*) *mod-add-left-eq mod-diff-left-eq of-nat-mod*)
thus *?thesis* **using** *c4-1* **by** (*metis asm4 diff-le-mono2 nat-le-linear numeral-le-iff*
verit-comp-simplify(*10*) *verit-comp-simplify*(*13*))
qed
hence ∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N=b+r*] (*mod m*) ∧ (*b = b1* ∨ *b = b2*)
if *asm4*:*N mod m = w+1* ∨ *N mod m = w+2* **for** *N* **using** *asm4* **by** *blast*
hence *c4*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N=b+r*] (*mod m*) ∧ (*b = b1* ∨ *b = b2*)
if *asm7*:*N mod m = k1−2* ∨ *N mod m = k1−1* **for** *N* **using** *w-cover asm7* **by** *auto*

have ∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N=b+r*] (*mod m*) ∧ (*b = b1* ∨ *b = b2*)
if *asm10*:*N mod m* ≥ *0* ∧ *N mod m* ≤*w* **for** *N* **using** *c2 c3 asm10*
using ‹⋀*N*. *0 < N mod m* ∧ *int* (*N mod m*) ≤ *w* ⟹ ∃ *b r*. *r* ≤ *m − 3* ∧ [*int N = b + int r*]
(*mod int m*) ∧ (*b = b1* ∨ *b = b2*)› **by** *blast*
hence *c5*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N=b+r*] (*mod m*) ∧ (*b = b1* ∨ *b = b2*)
if *asm11*:*N mod m* ≥ *0* ∧ *N mod m* ≤*k1−3* **for** *N* **using** *w-cover* **using** *asm11* **by** *force*

have *c6*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N=b+r*] (*mod m*) ∧ (*b = b1* ∨ *b = b2*)
if *asm9*:(*N mod m* ≥*0* ∧ *N mod m* ≤ *k1−3*) ∨ *N mod m = k1−2* ∨ *N mod m = k1−1* **for** *N*
using *c5 c4 asm9* **by** *blast*

hence *c7*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N=b+r*] (*mod m*) ∧ (*b = b1* ∨ *b = b2*)
if *asm12*:(*N mod m* ≥*0* ∧ *N mod m* ≤ *k1−3*) ∨ *N mod m = k1−2* ∨ *N mod m = k1−1* ∨
(*N mod m* ≥ *k1* ∧ *N mod m* ≤ *m−1*) **for** *N* **using** *asm12 c1* **by** *blast*

have ∀ *N*::*nat*. (*N mod m* ≥*0* ∧ *N mod m*≤ *k1−3*) ∨ *N mod m = k1−2* ∨ *N*

mod m = k1−1 ∨
(N mod m ≥ k1 ∧ N mod m ≤ m−1) **using** *k1-def*
by (*smt* (*verit*, *best*) *Suc-pred′ assms*(*1*) *bot-nat-0.extremum le-simps*(*2*)
mod-less-divisor
not-numeral-le-zero of-nat-0-less-iff of-nat-le-0-iff)

thus *?thesis* **using** *c7* **by** *auto*
qed

have *second*:∀ *N*::*nat*. ∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* =
b1 ∨ *b* = *b2*)
if *second-assum*:*b1 mod m* ≥*0* ∧ *b1 mod m* ≤*2*
proof −
have *case1*:∀ *N*::*nat*. ∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b*
= *b1* ∨ *b* = *b2*)
if *case1-assum*:*b1 mod m* = *0*
proof −
have ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N* = *b1*+*r*] (*mod m*) **if** *case1-1-assum*:*N mod*
m ≤*m−3* **for** *N*
using *case1-assum case1-1-assum*
by (*metis cong-add-rcancel-0 cong-mod-left cong-refl cong-sym-eq zmod-int*)
hence *case1-1*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1*
∨ *b* = *b2*)
if *case1-1-assum*:*N mod m* ≤*m−3* **for** *N* **using** *case1-1-assum* **by** *blast*

have [*N* = *b1*+(*m−2*)] (*mod m*) **if** *case1-2-assum*:*N mod m* = *m−2* **for** *N*
using *case1-2-assum*
case1-assum **by** (*metis* (*no-types*, *opaque-lifting*) *add.commute cong-add-lcancel-0*
cong-mod-right of-nat-mod unique-euclidean-semiring-class.cong-def)
hence [*N* = *b2*+(*m−4*)] (*mod m*) **if** *case1-2-assum*:*N mod m* = *m−2* **for** *N*
using *case1-2-assum assms*(*3*)
by (*smt* (*verit*, *best*) *add-leD2 assms*(*1*) *int-ops*(*2*) *numeral-Bit0 of-nat-diff*
of-nat-numeral
semiring-norm(*172*))
hence ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N* = *b2*+*r*] (*mod m*) **if** *case1-2-assum*:*N mod*
m = *m−2* **for** *N*
using *case1-2-assum*
by (*meson diff-le-mono2 less-num-simps*(*2*) *numeral-le-iff verit-comp-simplify*(*15*))
hence *case1-2*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m−3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1*
∨ *b* = *b2*)
if *case1-2-assum*:*N mod m* = *m−2* **for** *N* **using** *case1-2-assum* **by** *blast*

have [*N* = *b1*+(*m−1*)] (*mod m*) **if** *case1-3-assum*:*N mod m* = *m−1* **for** *N*
using *case1-3-assum*
case1-assum **by** (*metis* (*no-types*, *opaque-lifting*) *add.commute cong-add-lcancel-0*
cong-mod-right of-nat-mod unique-euclidean-semiring-class.cong-def)
hence [*N* = *b2*+(*m−3*)] (*mod m*) **if** *case1-3-assum*:*N mod m* = *m−1* **for** *N*
using *case1-3-assum assms*(*3*)
by (*smt* (*verit*, *best*) *assms*(*1*) *int-ops*(*2*) *int-ops*(*6*) *numeral-Bit0 nu-*

meral-Bit1 of-nat-mono
of-nat-numeral semiring-norm(172))
hence $\exists r$::*nat*. $(r \leq m-3) \wedge [N = b2+r]$ (*mod m*) **if** *case1-3-assum*:*N mod* $m = m-1$ **for** *N*
using *case1-3-assum* **by** *blast*
hence *case1-3*:$\exists b$::*int*. $\exists r$::*nat*. $(r \leq m-3) \wedge [N=b+r]$ (*mod m*) $\wedge$ $(b = b1 \vee b = b2)$
if *case1-3-assum*:*N mod* $m = m-1$ **for** *N* **using** *case1-3-assum* **by** *blast*

have $\forall N$::*nat*. $(N \bmod m = m-1) \vee (N \bmod m = m-2) \vee (N \bmod m \leq m-3)$
by (*smt* (*verit*, *ccfv-threshold*) *Suc-pred' assms*(*1*) *bot-nat-0.not-eq-extremum diff-diff-add*
diff-is-0-eq' le-simps(*2*) *mod-less-divisor nat-1-add-1 nat-less-le not-numeral-le-zero numeral.simps*(*3*) *semiring-norm*(*172*))
thus *?thesis* **using** *case1-1 case1-2 case1-3* **by** *blast*
qed

have *case2*:$\forall N$::*nat*. $\exists b$::*int*. $\exists r$::*nat*. $(r \leq m-3) \wedge [N=b+r]$ (*mod m*) $\wedge$ $(b = b1 \vee b = b2)$
if *case2-assum*:*b1 mod m* = *1*
proof −
have *case2b2*:*b2 mod m* = *3* **using** *case2-assum assms*(*3*) **by** (*smt* (*verit*) *assms*(*1*) *int-ops*(*2*)
mod-add-eq mod-pos-pos-trivial numeral-Bit0 of-nat-mono of-nat-numeral semiring-norm(*172*))

have $\exists r$::*nat*. $(r \leq m-3) \wedge [N = b2+r]$ (*mod m*) **if** *case2-1-assum*:*N mod m* $= m-1$ **for** *N*
proof −
have $[N = 3+(m-4)]$ (*mod m*) **using** *case2-1-assum*
by (*metis* (*mono-tags*, *lifting*) *Suc-eq-plus1 Suc-numeral add-diff-cancel-left*
arithmetic-simps(*1*) *arithmetic-simps*(*7*) *assms*(*1*)*mod-mod-trivial*
ordered-cancel-comm-monoid-diff-class.diff-add-assoc unique-euclidean-semiring-class.cong-def)
hence $[N = b2+(m-4)]$ (*mod m*) **using** *case2b2*
by (*metis* (*mono-tags*, *lifting*) *Num.of-nat-simps*(*4*) *mod-add-left-eq*
of-nat-mod of-nat-numeral unique-euclidean-semiring-class.cong-def)
thus *?thesis* **using** *le-diff-conv* **by** *fastforce*
qed
hence *case2-1*:$\exists b$::*int*. $\exists r$::*nat*. $(r \leq m-3) \wedge [N=b+r]$ (*mod m*) $\wedge$ $(b = b1 \vee b = b2)$
if *case2-1-assum*:*N mod* $m = m-1$ **for** *N* **using** *case2-1-assum* **by** *blast*

have $\exists r$::*nat*. $(r \leq m-3) \wedge [N = b2+r]$ (*mod m*) **if** *case2-2-assum*:*N mod m* $=0$ **for** *N*
proof −
have $(3+(m-3))$ *mod m* = *0* **using** *assms*(*1*) **by** *fastforce*
hence $(b2+(m-3))$ *mod m* = *0* **using** *case2b2* **by** (*metis Num.of-nat-simps*(*1*)
Num.of-nat-simps(*4*) *mod-add-left-eq of-nat-mod of-nat-numeral*)

thus *?thesis* **using** *case2-2-assum*
by (*metis int-ops*(*1*) *nat-le-linear of-nat-mod unique-euclidean-semiring-class.cong-def*)
qed
hence *case2-2*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m*−*3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1* ∨ *b* = *b2*)
if *case2-2-assum*:*N mod m* =*0* **for** *N* **using** *case2-2-assum* **by** *metis*

have ∃ *r*::*nat*. (*r* ≤ *m*−*3*) ∧ [*N* = *b1*+*r*] (*mod m*) **if**
case2-3-assum:*N mod m* ≤*m*−*2* ∧ *N mod m* ≥*1* **for** *N*
proof −
have ∃ *r*::*nat*. (*r*≤*m*−*3*)∧((*b1*+*r*) *mod m* = *l*) **if** *asml*:*l*≥*1* ∧ *l*≤*m*−*2* **for** *l*
proof −
define *r* **where** *r-def*:*r* = *l*−*1*
from *asml* **have** *r-range*:*r*≥*0* ∧ *r*≤*m*−*3* **using** *r-def* **by** *linarith*
have (*1*+*r*) *mod m* = *l* **using** *asml r-def r-range* **by** *fastforce*
hence (*b1*+*r*) *mod m* = *l* **using** *case2-assum*
by (*metis Num.of-nat-simps*(*3*) *int-ops*(*9*) *mod-add-left-eq plus-1-eq-Suc*)
thus *?thesis* **using** *asml r-range* **by** *blast*
qed
thus *?thesis* **using** *case2-3-assum*
by (*metis case2-3-assum of-nat-mod unique-euclidean-semiring-class.cong-def*)
qed
hence *case2-3*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m*−*3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1* ∨ *b* = *b2*)
if *case2-3-assum*:*N mod m* ≤*m*−*2* ∧ *N mod m* ≥*1* **for** *N* **using** *case2-3-assum*
by *blast*

have ∀ *N*::*nat*. *N mod m* = *0* ∨ (*N mod m* ≥*1* ∧ *N mod m* ≤*m*−*1*) **by** (*metis One-nat-def Suc-pred*
assms(*1*) *bot-nat-0.extremum-uniqueI leI less-Suc-eq-le mod-less-divisor not-numeral-le-zero*)
hence ∀ *N*::*nat*. *N mod m* = *0* ∨ (*N mod m* ≥*1* ∧ *N mod m* ≤*m*−*2*) ∨ *N mod m* = *m*−*1*
by (*smt* (*verit*) *arithmetic-simps*(*68*) *diff-diff-eq le-add-diff-inverse le-neq-implies-less le-simps*(*2*)
le-trans plus-1-eq-Suc)
thus *?thesis* **using** *case2-1 case2-2 case2-3* **by** (*metis* ‹⋀*N*. *N mod m* ≤ *m* − *2* ∧ *1* ≤ *N mod m*
⟹ ∃ *r*≤*m* − *3*. [*int N* = *b1* + *int r*] (*mod int m*)›)
qed

have *case3*:∀ *N*::*nat*. ∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m*−*3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1* ∨ *b* = *b2*)
if *case3-assum*:*b1 mod m* = *2*
proof −
have *case3b2*:*b2 mod m* = *4*
using *assms case3-assum*
by (*smt* (*verit*, *ccfv-SIG*) *Euclidean-Rings.pos-mod-sign dvd-mod-imp-dvd even-numeral int-ops*(*2*)
int-ops(*4*) *mod-diff-eq mod-pos-pos-trivial nat-1-add-1 numeral-Bit0*

of-nat-le-iff
of-nat-numeral plus-1-eq-Suc)

have $\exists r{::}nat.\ (r{\leq}m{-}3){\wedge}\ [N = b2{+}r]\ (mod\ m)$ **if** *case3-1-assum*:$N\ mod\ m = 0 \vee N\ mod\ m = 1$ **for** N
proof −
have $(4{+}(m{-}3))\ mod\ m = (4{+}m{-}3)\ mod\ m$ **using** *assms*(1) **by** *auto*
have $(4{+}m{-}3)\ mod\ m = (1{+}m)\ mod\ m$ **by** *simp*
hence $(4{+}(m{-}3))\ mod\ m = 1$ **using** ‹$(4{+}(m{-}3))\ mod\ m = (4{+}m{-}3)\ mod\ m$›
by (*smt* (*verit*, *best*) *Euclidean-Rings.pos-mod-bound add-lessD1 arith-special*(2) *assms*(1) *case3b2*
landau-product-preprocess(4) *mod-add-self2 mod-less numeral-Bit0 of-nat-numeral order-le-less*)
hence *caseone*:$(b2{+}(m{-}3))\ mod\ m = 1$ **using** *case3b2*
by (*metis Num.of-nat-simps*(2) *Num.of-nat-simps*(4) *mod-add-left-eq of-nat-mod of-nat-numeral*)

have $(4{+}(m{-}4))\ mod\ m = 0$ **using** *assms*(1) **by** *auto*
hence *casezero*:$(b2{+}(m{-}4))\ mod\ m = 0$ **using** *case3b2*
by (*metis* (*full-types*) *Num.of-nat-simps*(1) *Num.of-nat-simps*(4) *mod-add-left-eq of-nat-mod of-nat-numeral*)

show *?thesis* **using** *caseone casezero case3-1-assum*
by (*metis cong-int cong-mod-right cong-refl diff-le-mono2 nat-le-linear numeral-le-iff*
of-nat-0 of-nat-1 semiring-norm(69) *semiring-norm*(72))
qed
hence *case3-1*:$\exists b{::}int.\ \exists r{::}nat.\ (r \leq m{-}3) \wedge [N{=}b{+}r]\ (mod\ m) \wedge (b = b1 \vee b = b2)$
if *case3-1-assum*:$N\ mod\ m = 0 \vee N\ mod\ m = 1$ **for** N **using** *case3-1-assum*
by *metis*

have $\exists r{::}nat.(r{\leq}m{-}3){\wedge}\ [N = b1{+}r]\ (mod\ m)$ **if** *case3-2-assum*:$N\ mod\ m \geq 2 {\wedge} N\ mod\ m \leq m{-}1$ **for** N
proof −
have $\exists r{::}nat.\ (r{\leq}m{-}3){\wedge}((b1{+}r)\ mod\ m = l)$ **if** *asml1*:$l{\geq}2 \wedge l{\leq}m{-}1$ **for** l
proof −
define *r1* **where** *r1-def*:$r1 = l{-}2$
from *asml1* **have** *r1-range*:$r1{\geq}0 \wedge r1{\leq}m{-}2$ **using** *r1-def* **by** *linarith*
have $(2{+}r1)\ mod\ m = l$ **using** *asml1 r1-def r1-range* **by** *fastforce*
hence $(b1{+}r1)\ mod\ m = l$ **using** *case3-assum*
by (*metis Num.of-nat-simps*(4) *mod-add-left-eq of-nat-mod of-nat-numeral*)
thus *?thesis* **using** *asml1 r1-range* **by** (*metis One-nat-def diff-diff-add diff-le-mono*
nat-1-add-1 numeral-3-eq-3 plus-1-eq-Suc r1-def)
qed
thus *?thesis* **using** *case3-2-assum*

by (*metis case3-2-assum of-nat-mod unique-euclidean-semiring-class.cong-def*)
qed
hence *case3-2*:∃ *b*::*int*. ∃ *r*::*nat*. (*r* ≤ *m*−*3*) ∧ [*N*=*b*+*r*] (*mod m*) ∧ (*b* = *b1* ∨ *b* = *b2*)
if *case3-2-assum*:*N mod m* ≥*2*∧*N mod m* ≤ *m*−*1* **for** *N* **using** *case3-2-assum* **by** *blast*

have ∀ *N*::*nat*. *N mod m* = *0* ∨ (*N mod m* ≥*1* ∧ *N mod m* ≤*m*−*1*) **by** (*metis Suc-pred' assms*(*1*) *bot-nat-0.not-eq-extremum less-one mod-Suc-le-divisor rel-simps*(*76*) *verit-comp-simplify1*(*3*))
hence ∀ *N*::*nat*. *N mod m* = *0* ∨ *N mod m* = *1* ∨ (*N mod m* ≥*2* ∧ *N mod m* ≤*m*−*1*)
by (*metis Suc-eq-plus1 le-neq-implies-less le-simps*(*3*) *nat-1-add-1*)

thus *?thesis* **using** *case3-1 case3-2* **by** *blast*
qed

show *?thesis* **using** *case1 case2 case3* **using** *that* **by** *fastforce*
qed

show *?thesis* **using** *first second* **using** *assms*(*1*) **by** *force*
qed

The strong form of Cauchy's polygonal number theorem shows for a natural number N satisfying certain conditions, it may be written as the sum of $m+1$ polygonal numbers of order $m+2$, at most four of which are different from 0 or 1. This corresponds to Theorem 1.9 in [2].

theorem *Strong-Form-of-Cauchy-Polygonal-Number-Theorem-1*:
fixes *m N* :: *nat*
assumes *m*≥*4*
assumes *N*≥*108*∗*m*
shows ∃ *xs* :: *nat list*. (*length xs* = *m*+*1*) ∧ (*sum-list xs* = *N*) ∧ (∀ *k*≤*3*. ∃ *a*. *xs*! *k* = *polygonal-number m a*)
∧ (∀ *k* ∈ {*4*..*m*} . *xs*! *k* = *0* ∨ *xs*! *k* = *1*)

proof −
define *L* **where** *L-def*:*L* = (*2*/*3* + *sqrt* (*8*∗*N*/*m* − *8*)) − (*1*/*2* + *sqrt* (*6*∗*N*/*m* − *3*))
from *assms L-def* **have** *L*>*4* **using** *interval-length-greater-than-four*
apply(*rule-tac N* = *of-nat N* **and** *m* = *of-nat m* **in** *interval-length-greater-than-four*)
by *auto*
define *c1* **where** *c1-def*:*c1* = ⌈*1*/*2* + *sqrt* (*6*∗*N*/*m* − *3*)⌉
define *c2* **where** *c2-def*:*c2* = *c1*+*1*
define *c3* **where** *c3-def*:*c3* = *c1*+*2*
define *c4* **where** *c4-def*:*c4* = *c1*+*3*
from ‹*L*>*4*› *c1-def c2-def c3-def c4-def L-def* **have** *c4*<(*2*/*3* + *sqrt* (*8*∗*N*/*m* − *8*)) **by** *linarith*

have *N*/*m* ≥ *108* **using** *assms* **using** *le-divide-eq* **by** *fastforce*

hence *sqrt(6∗N/m − 3)≥1* **by** *simp*
hence *1/2 + sqrt(6∗N/m − 3) ≥1* **by** *linarith*
hence *c1 ≥1* **using** *c1-def* **by** *simp*

obtain *b1 b2* **where** *bproperties*:$\{b1, b2\} \subseteq \{c1, c2, c3, c4\} \wedge (b2 = b1+2) \wedge$ *odd b1*
using *two-consec-odd c1-def c2-def c3-def c4-def* **by** (*metis* (*no-types*, *opaque-lifting*) *Groups.add-ac*(*2*) *empty-subsetI even-plus-one-iff insert-commute insert-mono nat-arith.add1 numeral.simps*(*2*) *numeral.simps*(*3*))
have *b1andb2*:*odd b1 ∧ b2 = b1+2* **using** *bproperties* **by** *auto*
have *b1pos*:*b1 ≥1* **using** ‹*c1≥1*› *c2-def c3-def c4-def bproperties* **by** *auto*
hence *b2pos*:*b2 ≥3* **using** *bproperties* **by** *simp*
have *b2odd*:*odd b2* **using** *bproperties* **by** *simp*

obtain *b r* **where** *b-r*:$r \leq m-3 \wedge (b = b1 \vee b = b2) \wedge [int\ N = b+r]\ (mod\ m)$ **using** *b1andb2 assms*(*1*) *cong-classes* **by** *meson*
have *bpos*:*b≥1* **using** *b1pos b2pos b-r* **by** *auto*
have *bodd*:*odd b* **using** *b-r bproperties* **by** *auto*

define *a* **where** *a-def*:$a = b+2*(N-b-r)\ div\ m$
have *m-div-num*:*m dvd* $(N-b-r)$ **using** *b-r*
by (*simp add*: *diff-diff-add mod-eq-dvd-iff unique-euclidean-semiring-class.cong-def*)
hence $(N-b-r)/m = (N-b-r)\ div\ m$ **by** (*simp add*: *real-of-int-div*)
hence *a-def1*:$a = b+2*(N-b-r)/m$ **using** *a-def* **by** (*metis* ‹*int m dvd int N − b − int r*› *dvd-add-right-iff mult-2 of-int-add of-int-of-nat-eq real-of-int-div*)
have *N−m>0* **using** *assms* **by** *linarith*
hence *N−r>0* **using** *b-r* **by** *force*
hence $(N-b-r) = (N-r)-b$ **by** *linarith*
hence $(N-b-r)/m = (N-r)/m - b/m$ **by** (*metis diff-divide-distrib int-of-reals*(*3*) *of-int-of-nat-eq*)
hence $a = b+2*((N-r)/m - b/m)$ **using** *a-def1* **by** (*metis int-of-reals*(*6*) *of-int-mult times-divide-eq-right*)
hence *a-def2*:$a = b- b*2/m+2*(N-r)/m$ **by** *simp*
have $b*(1-2/m) = b*1-b*(2/m)$ **by** (*simp add*: *Rings.ring-distribs*(*4*))
hence *a-def3*:$a = b*(1-2/m) + 2*(N-r)/m$ **using** *a-def2* **by** *simp*
have *1−2/m>0* **using** *assms*(*1*) **by** *simp*
hence *size1*:$b*(1-2/m)>0$ **using** *bpos* **by** *simp*
have *N−r>0* **using** *b-r assms* **by** *auto*
hence *size2*:$2*(N-r)/m>0$ **using** *assms*(*1*) **by** *simp*
have *apos*:*a≥1* **using** *size1 size2 a-def3* **by** *simp*

have *odd* $(b+2*(N-b-r)\ div\ m)$ **using** *m-div-num b-r b2odd bproperties*
by (*metis div-mult-swap zdvd-reduce*)
hence *aodd*:*odd a* **using** *a-def* **by** *simp*

from *a-def1* **have** $a-b = 2*(N-b-r)/m$ **by** *simp*

hence $m*(a-b)/2 = N-b-r$ **using** *assms(1)* **by** *simp*
hence *N-expr*:$N = r+b+m*(a-b)/2$ **by** *simp*

have $b1 \geq c1$ **using** *bproperties c2-def c3-def c4-def* **by** *force*
hence $b1 \geq 1/2 + sqrt\ (6*N/m - 3)$ **using** *c1-def* **using** *ceiling-le-iff* **by** *blast*
have *b-ineq1*:$b \geq 1/2 + sqrt\ (6*N/m - 3)$ **using** *b-r bproperties*
using ‹*1 / 2 + sqrt (real (6 * N) / real m − 3) ≤ real-of-int b1*› **by** *fastforce*

have $b2 \leq c4$ **using** *bproperties c1-def c2-def c3-def c4-def* **by** *fastforce*
hence $b2 \leq (2/3 + sqrt\ (8*N/m - 8))$
using ‹*real-of-int c4 < 2 / 3 + sqrt (real (8 * N) / real m − 8)*› **by** *linarith*
hence *b-ineq2*:$b \leq (2/3 + sqrt\ (8*N/m - 8))$ **using** *b-r bproperties* **by** *linarith*

define *Nr* **where** $Nr = real\text{-}of\text{-}nat\ N$
define *mr* **where** $mr = real\ m$
define *ar* **where** $ar = real\text{-}of\text{-}int\ a$
define *br* **where** $br = real\text{-}of\text{-}int\ b$
define *rr* **where** $rr = real\text{-}of\text{-}nat\ r$
from *assms(1)* **have** $mr \geq 3$ **using** *mr-def* **by** *auto*
from *assms(2)* **have** $N \geq 2*m$ **by** *simp*
hence $Nr \geq 2*mr$ **using** *Nr-def mr-def* ‹$N \geq 2 * m$› **by** *auto*
moreover have $br \geq 0$ **using** *br-def bpos* **by** *auto*
moreover have $mr \geq 3$ **using** *mr-def assms* **by** *auto*
moreover have $ar \geq 0$ **using** *ar-def apos* **by** *auto*
moreover have $rr \geq 0$ **using** *rr-def b-r* **by** *auto*
moreover have $mr > rr$ **using** *mr-def rr-def b-r assms(1)* **by** *linarith*
moreover have $Nr = mr*(ar-br)/2+br+rr$ **using** *Nr-def mr-def ar-def br-def N-expr rr-def* **by** *auto*
moreover have $1/2+sqrt(6*Nr/mr-3) \leq br \wedge br \leq 2/3+sqrt(8*Nr/mr-8)$ **using** *Nr-def mr-def br-def b-ineq1 b-ineq2* **by** *auto*
ultimately have $br\hat{}2 < 4*ar \wedge 3*ar < br\hat{}2+2*br+4$ **using** *Cauchy-lemma* **by** *auto*
hence *real-ineq*:$(real\text{-}of\text{-}int\ b)\hat{}2 < 4*(real\text{-}of\text{-}int\ a) \wedge 3*(real\text{-}of\text{-}int\ a) < (real\text{-}of\text{-}int\ b)\hat{}2 + 2*(real\text{-}of\text{-}int\ b) + 4$
using *br-def ar-def* **by** *auto*
hence *int-ineq1*: $b\hat{}2 < 4*a$ **using** *of-int-less-iff* **by** *fastforce*
from *real-ineq* **have** *int-ineq2*: $3*a < b\hat{}2+2*b+4$ **using** *of-int-less-iff* **by** *fastforce*

have *con1*:$nat\ a \geq 1$ **using** *apos* **by** *auto*
have *con2*:$nat\ b \geq 1$ **using** *bpos* **by** *auto*
have *con3*:$odd\ (nat\ a)$ **using** *aodd apos even-nat-iff* **by** *auto*
have *con4*:$odd\ (nat\ b)$ **using** *bodd bpos even-nat-iff* **by** *auto*
have $(nat\ b)\hat{}2 = b\hat{}2$ **using** ‹*nat b ≥1*› **by** *auto*
hence *con5*:$(nat\ b)\hat{}2 < 4*(nat\ a)$ **using** *int-ineq1* **by** *linarith*
have *con6*:$3*(nat\ a) < (nat\ b)\hat{}2+2*(nat\ b)+4$ **using** ‹$(nat\ b)\hat{}2 = b\hat{}2$› *int-ineq2* **by** *linarith*
obtain *s t u v* **where** *stuv*:$s \geq 0 \wedge t \geq 0 \wedge u \geq 0 \wedge v \geq 0 \wedge int(nat\ a) = s\hat{}2 + t\hat{}2 + u\hat{}2 + v\hat{}2 \wedge$

int(*nat b*) = *s*+*t*+*u*+*v* **using** *four-nonneg-int-sum con1 con2 con3 con4 con5 con6* **by** *presburger*
have *a-expr*:*a* = *s*^2 + *t*^2 + *u*^2 + *v*^2 **using** *apos stuv* **by** *linarith*
have *b-expr*:*b* = *s*+*t*+*u*+*v* **using** *bpos stuv* **by** *linarith*

from *N-expr* **have** *N* = *m*/2∗(*s*^2−*s*+*t*^2−*t*+*u*^2−*u*+*v*^2−*v*)+*r*+(*s*+*t*+*u*+*v*) **using** *a-expr b-expr* **by** *simp*
hence *N-expr2*:*N* = *m*/2∗(*s*^2−*s*)+ *m*/2∗(*t*^2−*t*)+ *m*/2∗(*u*^2−*u*)+ *m*/2∗(*v*^2−*v*)+ *r*+(*s*+*t*+*u*+*v*)
by (*metis* (*no-types*, *opaque-lifting*) *add-diff-eq nat-distrib*(2) *of-int-add*)
have *s-div2*:*m*/2∗(*s*^2−*s*) = *m*∗(*s*^2−*s*) *div* 2 **using** *real-of-int-div* **by** *auto*
have *t-div2*:*m*/2∗(*t*^2−*t*) = *m*∗(*t*^2−*t*) *div* 2 **using** *real-of-int-div* **by** *auto*
have *u-div2*:*m*/2∗(*u*^2−*u*) = *m*∗(*u*^2−*u*) *div* 2 **using** *real-of-int-div* **by** *auto*
have *v-div2*:*m*/2∗(*v*^2−*v*) = *m*∗(*v*^2−*v*) *div* 2 **using** *real-of-int-div* **by** *auto*
have *N-expr3*:*N* = (*m*∗(*s*^2−*s*) *div* 2+*s*)+(*m*∗(*t*^2−*t*) *div* 2+*t*)+(*m*∗(*u*^2−*u*) *div* 2+*u*)+(*m*∗(*v*^2−*v*) *div* 2+*v*)+*r*
using *s-div2 t-div2 u-div2 v-div2 N-expr2* **by** *simp*

define *sn* **where** *sn* = *nat s*
define *tn* **where** *tn* = *nat t*
define *un* **where** *un* = *nat u*
define *vn* **where** *vn* = *nat v*
have *seqsn*:*s*^2−*s* = *sn*^2 − *sn* **using** *stuv sn-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)
have *teqtn*:*t*^2−*t* = *tn*^2 − *tn* **using** *stuv tn-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)
have *uequn*:*u*^2−*u* = *un*^2 − *un* **using** *stuv un-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)
have *veqvn*:*v*^2−*v* = *vn*^2 − *vn* **using** *stuv vn-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)

from *N-expr3* **have**
N = (*m*∗(*sn*^2−*sn*) *div* 2+*s*)+(*m*∗(*tn*^2−*tn*) *div* 2+*t*)+(*m*∗(*un*^2−*un*) *div* 2+*u*)+(*m*∗(*vn*^2−*vn*) *div* 2+ *v*)+*r*
using *seqsn teqtn uequn veqvn* **by** (*metis* (*mono-tags*, *lifting*) *int-ops*(2) *int-ops*(4) *int-ops*(7)
numeral-Bit0 numeral-code(1) *plus-1-eq-Suc zdiv-int*)
hence *N* = (*m*∗(*sn*^2−*sn*) *div* 2+*sn*)+(*m*∗(*tn*^2−*tn*) *div* 2+*tn*)+(*m*∗(*un*^2−*un*) *div* 2+*un*)+(*m*∗(*vn*^2−*vn*) *div* 2+ *v*)+*r*
using *sn-def tn-def un-def stuv int-nat-eq int-ops*(5) **by** *presburger*
hence *N* = (*m*∗(*sn*^2−*sn*) *div* 2+*sn*)+(*m*∗(*tn*^2−*tn*) *div* 2+*tn*)+(*m*∗(*un*^2−*un*) *div* 2+*un*)+(*m*∗(*vn*^2−*vn*) *div* 2+ *vn*)+*r*
using *vn-def stuv* **by** (*smt* (*verit*, *del-insts*) *Num.of-nat-simps*(4) *int-nat-eq of-nat-eq-iff*)
hence *N* = (*m*∗ *sn*∗(*sn*−1) *div* 2+*sn*)+(*m*∗*tn*∗(*tn*−1) *div* 2+*tn*)+(*m*∗*un*∗(*un*−1) *div* 2+*un*)+(*m*∗ *vn*∗(*vn*−1) *div* 2+ *vn*)+*r*
by (*smt* (*verit*, *ccfv-threshold*) *more-arith-simps*(11) *mult.right-neutral power2-eq-square right-diff-distrib*′)
hence *N-expr4*:*N* = *polygonal-number m sn* + *polygonal-number m tn* + *polyg-*

onal-number m un + polygonal-number m vn +r
using *Polygonal-Number-Theorem-Gauss.polygonal-number-def* **by** *presburger*

define *T* **where** *T-def*:*T* = [*polygonal-number m sn*,*polygonal-number m tn*,*polygonal-number m un*,*polygonal-number m vn*]
define *ones* **where** *ones-def*:*ones* = *replicate r* (*1*::*nat*)
define *zeros* **where** *zeros-def*:*zeros* = *replicate* (*m*+*1*−*4*−*r*) (*0*::*nat*)
define *final* **where** *final-def*:*final* = *T*@*ones*@*zeros*

have *m*+*1*−*4*−*r*≥*0* **using** *assms*(*1*) *b-r* **by** *force*
hence *4*+*r*+(*m*+*1*−*4*−*r*) = *m*+*1* **using** *assms*(*1*) *b-r* **by** *force*
have *length final* = *4*+*r*+(*m*+*1*−*4*−*r*) **using** *final-def T-def ones-def zeros-def* **by** *auto*
hence *final-length*:*length final* = *m*+*1* **using** ‹*4*+*r*+(*m*+*1*−*4*−*r*) = *m*+*1*› **by** *simp*
have *T-sum*:*sum-list T* = *polygonal-number m sn* + *polygonal-number m tn* + *polygonal-number m un* + *polygonal-number m vn* **by** (*simp add*: *T-def*)
have *ones-sum*:*sum-list ones* = *r* **using** *ones-def* **by** (*simp add*: *sum-list-replicate*)
have *zeros-sum*:*sum-list zeros* = *0* **using** *zeros-def* **by** *simp*
have *sum-list final* = *sum-list T* + *sum-list ones* + *sum-list zeros* **using** *final-def* **by** *simp*
hence *final-sum*:*sum-list final* = *N* **using** *N-expr4* **by** (*simp add*: *T-sum ones-sum zeros-sum*)

have *final-0th*:*final*! *0* = *polygonal-number m sn* **using** *final-def T-def* **by** *simp*
have *final-1st*:*final*! *1* = *polygonal-number m tn* **using** *final-def T-def* **by** *simp*
have *final-2nd*:*final*! *2* = *polygonal-number m un* **using** *final-def T-def* **by** *simp*
have *final-3rd*:*final*! *3* = *polygonal-number m vn* **using** *final-def T-def* **by** *simp*

have *first-four*:∀ *k*≤*3*. ∃ *a*. *final*! *k* = *polygonal-number m a* **using** *final-0th final-1st final-2nd final-3rd*
by (*metis Suc-eq-plus1 add-leD2 arith-simps*(*50*) *le-simps*(*2*) *numeral-Bit0 numeral-Bit1*
numeral-One verit-comp-simplify1(*3*) *verit-la-disequality*)

have *length T* = *4* **using** *T-def* **by** *simp*
have ∀ *k*<*length* (*ones*@*zeros*). (*ones*@*zeros*)! *k* =*1* ∨ (*ones*@*zeros*)! *k* =*0* **using** *ones-def zeros-def*
by (*simp add*: *nth-append*)
hence *final*! *k* = *1* ∨ *final*! *k* = *0* **if** *k*≥*4* ∧ *k*<(*length final*) **for** *k*
using ‹*length T* = *4*› *final-def that* **by** (*metis add-less-cancel-left le-add-diff-inverse*
length-append nth-append verit-comp-simplify1(*3*))
hence *other-terms*:∀ *k* ∈ {*4*..*m*} . *final*! *k* = *0* ∨ *final*! *k* = *1* **using** *final-length*
by (*metis Suc-eq-plus1 atLeastAtMost-iff le-simps*(*2*))

show *?thesis* **using** *final-length final-sum first-four other-terms* **by** *auto*
qed

theorem *Strong-Form-of-Cauchy-Polygonal-Number-Theorem-2*:
fixes *N* :: *nat*
assumes *N*≥*324*
shows ∃ *p1 p2 p3 p4 r* ::*nat*. *N* = *p1*+*p2*+*p3*+*p4*+*r* ∧ (∃ *k1*. *p1* = *polygonal-number 3 k1*) ∧ (∃ *k2*. *p2* = *polygonal-number 3 k2*)
∧ (∃ *k3*. *p3* = *polygonal-number 3 k3*) ∧ (∃ *k4*. *p4* = *polygonal-number 3 k4*) ∧ (*r* = *0* ∨ *r* = *1*)

proof −
define *L* **where** *L-def*:*L* = (*2*/*3* + *sqrt* (*8*∗*N*/*3* − *8*)) − (*1*/*2* + *sqrt* (*6*∗*N*/*3* − *3*))
from *assms L-def* **have** *L*>*4* **using** *interval-length-greater-than-four*
apply −
apply(*rule interval-length-greater-than-four*[**where** *N* = *of-nat N* **and** *m* = *of-nat 3*])
by *auto*
define *c1* **where** *c1-def*:*c1* = ⌈*1*/*2* + *sqrt* (*6*∗*N*/*3* − *3*)⌉
define *c2* **where** *c2-def*:*c2* = *c1*+*1*
define *c3* **where** *c3-def*:*c3* = *c1*+*2*
define *c4* **where** *c4-def*:*c4* = *c1*+*3*
from ‹*L*>*4*› *c1-def c2-def c3-def c4-def L-def* **have** *c4*<(*2*/*3* + *sqrt* (*8*∗*N*/*3* − *8*)) **by** *linarith*

define *Nn* **where** *Nn* = *int N*
have *c4*<(*2*/*3* + *sqrt* (*8*∗*Nn*/*3* − *8*)) **using** *Nn-def* ‹*c4*<(*2*/*3* + *sqrt* (*8*∗*N*/*3* − *8*))› **by** *simp*
have *Nn3*:(*Nn*−*3*)^*2* − (*sqrt* (*8*∗*Nn*/*3* − *8*))^*2* = *Nn*^*2*−*3*∗*Nn*−*3*∗*Nn*+*9* − (*sqrt* (*8*∗*Nn*/*3* − *8*))^*2*
using *assms Nn-def power2-diff* **by** (*simp add*: *power2-eq-square algebra-simps*)
have (*Nn*−*3*)^*2* − (*sqrt* (*8*∗*Nn*/*3* − *8*))^*2* = *Nn*^*2*−*3*∗*Nn*−*3*∗*Nn*+*9* − (*8*∗*Nn*/*3* − *8*) **using** *assms Nn-def Nn3* **by** *fastforce*
hence (*Nn*−*3*)^*2* − (*sqrt* (*8*∗*Nn*/*3* − *8*))^*2* = *Nn*^*2*−*6*∗*Nn*+*9*−*8*∗*Nn*/*3* +*8* **by** *linarith*
hence *Nn4*:(*Nn*−*3*)^*2* − (*sqrt* (*8*∗*Nn*/*3* − *8*))^*2* = *Nn*∗(*Nn*−*26*/*3*)+*17* **by** (*simp add*: *Rings.ring-distribs*(*4*) *power2-eq-square*)
have *Nn*∗(*Nn*−*26*/*3*)+*17*>*17* **using** *assms Nn-def* **by** *auto*
hence (*Nn*−*3*)^*2* − (*sqrt* (*8*∗*Nn*/*3* − *8*))^*2* > *0* **using** *Nn4* **by** *auto*
hence *Nn*−*3* > *sqrt* (*8*∗*Nn*/*3* − *8*) **using** *assms Nn-def* **by** (*simp add*: *real-less-lsqrt*)
hence *Nn*−*2* > *sqrt* (*8*∗*Nn*/*3* − *8*)+*2*/*3* **by** *linarith*
hence *N* > *c4* **using** *Nn-def* ‹*c4*<(*2*/*3* + *sqrt* (*8*∗*Nn*/*3* − *8*))› **by** *simp*

have *N*/*3* ≥ *108* **using** *assms* **using** *le-divide-eq* **by** *fastforce*
hence *sqrt*(*6*∗*N*/*3* − *3*)≥*1* **by** *simp*
hence *1*/*2* + *sqrt*(*6*∗*N*/*3* − *3*) ≥*1* **by** *linarith*
hence *c1* ≥*1* **using** *c1-def* **by** *simp*

obtain *b1 b2* **where** *bproperties*:{*b1*, *b2*} ⊆ {*c1*, *c2*, *c3*, *c4*} ∧ (*b2* = *b1*+*2*) ∧ *odd b1*
using *two-consec-odd c1-def c2-def c3-def c4-def* **by** (*metis* (*no-types*, *opaque-lifting*)

Groups.add-ac(2)
empty-subsetI even-plus-one-iff insert-commute insert-mono nat-arith.add1 numeral.simps(2)
numeral.simps(3))
have *b1andb2:odd b1 ∧ b2 = b1+2* **using** *bproperties* **by** *auto*
have *b1pos:b1 ≥1* **using** ‹*c1≥1*› *c2-def c3-def c4-def bproperties* **by** *auto*
hence *b2pos:b2 ≥3* **using** *bproperties* **by** *simp*
have *b2odd:odd b2* **using** *bproperties* **by** *simp*
define *b1n* **where** *b1n = nat b1*
define *b2n* **where** *b2n = nat b2*

from *b1n-def b1pos* **have** *b1n mod 3 = b1 mod 3* **using** *int-ops(9)* **by** *force*
from *b2n-def b2pos* **have** *b2n mod 3 = b2 mod 3* **using** *int-ops(9)* **by** *force*

have *b-and-r:∃ b r::nat. [N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0 ∨ r = 1)*
proof −
have *case1:∃ b r::nat. [N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0 ∨ r = 1)*
if *asm1:b1 mod 3 = 0*
proof −
have *b1n mod 3 = 0* **using** *asm1* ‹*b1n mod 3 = b1 mod 3*› **by** *simp*
hence *b2n mod 3 = 2* **using** ‹*b2n mod 3 = b2 mod 3*› *bproperties asm1* **by** *fastforce*
have *case1-1:[0 = b1n+0] (mod 3)* **using** ‹*b1n mod 3 = 0*›
by (*metis mod-0 nat-arith.rule0 unique-euclidean-semiring-class.cong-def*)
have *case1-2:[1 = b1n+1] (mod 3)* **using** ‹*b1n mod 3 = 0*›
by (*metis* ‹*[0 = b1n + 0] (mod 3)*› *add.commute cong-add-lcancel-0-nat cong-sym*)
have *case1-3:[2 = b2n+0] (mod 3)* **using** ‹*b2n mod 3 = 2*›
by (*simp add*: *unique-euclidean-semiring-class.cong-def*)
have *∀ N::nat. N mod 3 = 0 ∨ N mod 3 ≥ 1* **by** *linarith*
hence *∀ N::nat. N mod 3 = 0 ∨ N mod 3 = 1 ∨ N mod 3 = 2* **by** *linarith*
hence *∀ N. ∃ b r::nat. [N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0 ∨ r = 1)*
if *asm1:b1 mod 3 = 0* **using** *case1-1 case1-2 case1-3* **by** (*metis cong-mod-left*)
thus *?thesis* **using** *asm1* **by** *auto*
qed

have *case2:∃ b r::nat. [N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0 ∨ r = 1)*
if *asm2:b1 mod 3 = 1*
proof −
have *b1n mod 3 = 1* **using** *asm2* ‹*b1n mod 3 = b1 mod 3*› **by** *simp*
hence *b2n mod 3 = 0* **using** ‹*b2n mod 3 = b2 mod 3*› *bproperties asm2*
by (*smt* (*verit*, *best*) *Euclidean-Rings.pos-mod-bound Euclidean-Rings.pos-mod-sign int-ops(1) mod-diff-eq mod-pos-pos-trivial of-nat-eq-iff*)
have *case2-1:[0 = b2n+0] (mod 3)* **using** ‹*b2n mod 3 = 0*›
by (*metis mod-0 nat-arith.rule0 unique-euclidean-semiring-class.cong-def*)
have *case2-2:[1 = b1n+0] (mod 3)* **using** ‹*b1n mod 3 = 1*›

```
    by (simp add: unique-euclidean-semiring-class.cong-def)
   have case2-3:[2 = b1n+1] (mod 3) using ‹b1n mod 3 = 1›
    by (metis case2-2 cong-add-rcancel-nat nat-1-add-1 nat-arith.rule0)
   have ∀ N::nat. N mod 3 = 0 ∨ N mod 3 ≥ 1 by linarith
   hence ∀ N::nat. N mod 3 = 0 ∨ N mod 3 = 1 ∨ N mod 3 = 2 by linarith
   hence ∀ N. ∃ b r::nat. [N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0
∨ r = 1)
   if asm2:b1 mod 3 = 1 using case2-1 case2-2 case2-3 by (metis cong-mod-left)
   thus ?thesis using asm2 by auto
  qed

  have case3:∃ b r::nat. [N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0 ∨
r = 1)
   if asm3:b1 mod 3 = 2
  proof −
   have b1n mod 3 = 2 using asm3 ‹b1n mod 3 = b1 mod 3› by simp
   have (b1+2) mod 3 = (2+2) mod 3 using asm3 by (metis Groups.add-ac(2)
mod-add-right-eq)
   hence b2n mod 3 = 1 using ‹b2n mod 3 = b2 mod 3› bproperties by simp
   have case3-1:[0 = b1n+1] (mod 3) using ‹b1n mod 3 = 2›
       by (metis One-nat-def add.commute mod-0 mod-add-right-eq mod-self
nat-1-add-1 numeral-3-eq-3
       plus-1-eq-Suc unique-euclidean-semiring-class.cong-def)
   have case3-2:[1 = b2n+0] (mod 3) using ‹b2n mod 3 = 1›
    by (simp add: unique-euclidean-semiring-class.cong-def)
   have case3-3:[2 = b1n+0] (mod 3) using ‹b1n mod 3 = 2›
    by (simp add: unique-euclidean-semiring-class.cong-def)
   have ∀ N::nat. N mod 3 = 0 ∨ N mod 3 ≥ 1 by linarith
   hence ∀ N::nat. N mod 3 = 0 ∨ N mod 3 = 1 ∨ N mod 3 = 2 by linarith
   hence ∀ N. ∃ b r::nat. [N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0
∨ r = 1)
   if asm3:b1 mod 3 = 2 using case3-1 case3-2 case3-3 by (metis cong-mod-left)
   thus ?thesis using asm3 by auto
  qed

  have b1 mod 3 = 0 ∨ b1 mod 3 = 1 ∨ b1 mod 3 = 2 by auto
  thus ?thesis using case1 case2 case3 by auto
 qed

 obtain b r where b-r:[N = b+r] (mod 3) ∧ (b = b1n ∨ b = b2n) ∧ (r = 0 ∨
r = 1)
  using b-and-r by auto

 have bpos:b≥1 using b1pos b2pos b-r b1n-def b2n-def by auto
 have bodd:odd b
  using b-r bproperties by (metis b1n-def b2n-def b2odd bpos even-nat-iff nat-eq-iff2
rel-simps(45))

 define a where a-def:a = b+2∗(N−b−r) div 3
```

have *int b1n = b1* **using** *b1n-def b1pos* **by** *linarith*
have *int b2n = b2* **using** *b2n-def b2pos* **by** *linarith*
have *m-div-num:3 dvd (N−b−r)* **using** *b-r*
by (*metis cong-altdef-nat diff-diff-left diff-is-0-eq' dvd-0-right nat-le-linear*)
hence *a-def1:a = b+2∗(N−b−r)/3* **using** *a-def m-div-num real-of-nat-div* **by** *auto*
from ‹*N>c4*› **have** *N>b* **using** *b-r bproperties b1n-def b2n-def*
by (*smt* (*verit*, *del-insts*) ‹*int b1n = b1*› ‹*int b2n = b2*› *c2-def c3-def c4-def empty-iff insert-iff insert-subset of-nat-less-imp-less*)
hence *(N−b−r)/3 = (N−r)/3 − b/3* **using** ‹*b < N*› *b-r* **by** *force*
hence *a = b− b∗2/3+2∗(N−r)/3* **using** *a-def1* **by** *linarith*
hence *a-def3:a = b∗(1−2/3) + 2∗(N−r)/3* **by** *simp*

have *size1:b∗(1−2/3)>0* **using** *bpos* **by** *simp*
have *N−r>0* **using** *b-r assms* **by** *auto*
hence *size2:2∗(N−r)/3>0* **using** *assms(1)* **by** *simp*
have *apos:a≥1* **using** *size1 size2 a-def3* **by** *simp*

have *odd (b+2∗(N−b−r) div 3)* **using** *m-div-num b-r b2odd bproperties* **by** (*simp add*: *bodd mult-2*)
hence *aodd:odd a* **using** *a-def* **by** *simp*
from *a-def1* **have** *a−b = 2∗(N−b−r)/3* **by** *simp*
hence *(a−b)/2 = (N−b−r)/3* **by** *simp*
hence *3∗(a−b)/2 = N−b−r* **by** *simp*
have *N−b−r≥0* **using** *b-r* **by** *simp*
hence *N-expr:N = r+b+3∗(a−b)/2* **using** ‹*N−b−r≥0*› ‹*b < N*› *b-r* ‹*real (3 ∗ (a − b)) / 2 = real (N − b − r)*› **by** *linarith*
from *a-def* ‹*N−b−r≥0*› **have** *a≥b* **using** *a-def le-add1* **by** *blast*

have *b1 ≥ c1* **using** *bproperties c2-def c3-def c4-def* **by** *force*
hence *b1 ≥ 1/2 + sqrt (6∗N/3 − 3)* **using** *c1-def* **using** *ceiling-le-iff* **by** *blast*
hence *b1ngreater:b1n ≥ 1/2 + sqrt (6∗N/3 − 3)* **using** *b1n-def* **by** *simp*
hence *b2ngreater:b2n ≥ 1/2 + sqrt (6∗N/3 − 3)* **using** *bproperties b1n-def b2n-def* **by** *linarith*
hence *b-ineq1:b ≥ 1/2 + sqrt (6∗N/3 − 3)* **using** *b-r b1ngreater* **by** *auto*

have *b2 ≤ c4* **using** *bproperties c1-def c2-def c3-def c4-def* **by** *fastforce*
hence *b2 ≤ (2/3 + sqrt (8∗N/3 − 8))*
using ‹*real-of-int c4 < 2 / 3 + sqrt (real (8 ∗ N) / 3 − 8)*› **by** *linarith*
hence *b2nsmaller:b2n ≤ (2/3 + sqrt (8∗N/3 − 8))* **using** *b2n-def* **by** (*metis* ‹*int b2n = b2*› *of-int-of-nat-eq*)
hence *b1n ≤ (2/3 + sqrt (8∗N/3 − 8))* **using** *b1n-def bproperties* **using** ‹*int b2n = b2*› **by** *linarith*
hence *b-ineq2:b≤(2/3 + sqrt (8∗N/3 − 8))* **using** *b-r b2nsmaller* **by** *auto*

define *Nr* **where** *Nr = real-of-nat N*
define *ar* **where** *ar = real-of-int a*
define *br* **where** *br = real-of-int b*
define *rr* **where** *rr = real-of-nat r*

define *m* **where** *m = real-of-nat 3*
from *assms* **have** *N≥2∗m* **using** *m-def* **by** *simp*
then have *Nr ≥ 2∗m* **using** *Nr-def ‹N ≥ 2 ∗ m›* **by** *auto*
moreover have *br≥0* **using** *br-def bpos* **by** *auto*
moreover have *ar≥0* **using** *ar-def apos* **by** *auto*
moreover have *rr≥0* **using** *rr-def b-r* **by** *auto*
moreover have *m≥3* **using** *m-def* **by** *auto*
moreover have *m>rr* **using** *m-def rr-def b-r* **by** *auto*
moreover have *Nr = m∗(ar−br)/2+br+rr* **using** *Nr-def ar-def br-def N-expr rr-def m-def ‹a≥b›* **by** *auto*
moreover have *1/2+sqrt(6∗Nr/m−3)≤br ∧ br≤2/3+sqrt(8∗Nr/m−8)* **using** *Nr-def br-def b-ineq1 b-ineq2 m-def* **by** *auto*
ultimately have *br^2<4∗ar ∧ 3∗ar<br^2+2∗br+4* **using** *Cauchy-lemma* **by** *auto*
hence *real-ineq*:*(real-of-int b)^2 < 4∗(real-of-int a) ∧ 3∗(real-of-int a) < (real-of-int b)^2 + 2∗(real-of-int b) + 4*
using *br-def ar-def* **by** *auto*
hence *nat-ineq1*: *b^2<4∗a* **using** *br-def* **by** (*smt* (*verit*, *del-insts*) *Num.of-nat-simps*(*4*) *mult.commute mult-2-right nat-distrib*(*1*) *numeral-Bit0 of-int-of-nat-eq of-nat-less-of-nat-power-cancel-iff*)
from *real-ineq* **have** *nat-ineq2*: *3∗a<b^2+2∗b+4* **using** *ar-def br-def of-nat-less-iff* **by** *fastforce*

obtain *s t u v* **where** *stuv*:*s ≥ 0 ∧ t ≥ 0 ∧ u ≥ 0 ∧ v ≥ 0 ∧ int a = s^2 + t^2 + u^2 + v^2 ∧*
int b = s+t+u+v **using** *apos bpos aodd bodd nat-ineq1 nat-ineq2 four-nonneg-int-sum* **by** *presburger*
have *a-expr*:*a = s^2 + t^2 + u^2 + v^2* **using** *apos stuv* **by** *linarith*
have *b-expr*:*b = s+t+u+v* **using** *bpos stuv* **by** *linarith*

have *N = r + (s+t+u+v)+ 3∗(a−(s+t+u+v))/2* **using** *b-expr N-expr*
by (*metis Num.of-nat-simps*(*4*) *Num.of-nat-simps*(*5*) *‹b ≤ a› of-int-of-nat-eq of-nat-diff of-nat-numeral*)
hence *N = 3/2∗(s^2−s+t^2−t+u^2−u+v^2−v)+r+(s+t+u+v)* **using** *a-expr* **by** *simp*
hence *N-expr2*:*N = 3/2∗(s^2−s)+ 3/2∗(t^2−t)+ 3/2∗(u^2−u)+ 3/2∗(v^2−v)+ r+(s+t+u+v)*
by (*metis* (*no-types*, *opaque-lifting*) *add-diff-eq nat-distrib*(*2*) *of-int-add*)

have *s-div2*:*3/2∗(s^2−s) = 3∗(s^2−s) div 2* **using** *real-of-int-div* **by** *auto*
have *t-div2*:*3/2∗(t^2−t) = 3∗(t^2−t) div 2* **using** *real-of-int-div* **by** *auto*
have *u-div2*:*3/2∗(u^2−u) = 3∗(u^2−u) div 2* **using** *real-of-int-div* **by** *auto*
have *v-div2*:*3/2∗(v^2−v) = 3∗(v^2−v) div 2* **using** *real-of-int-div* **by** *auto*
have *N-expr3*:*N = (3∗(s^2−s) div 2+s)+(3∗(t^2−t) div 2+t)+(3∗(u^2−u) div 2+u)+(3∗(v^2−v) div 2+v)+r*
using *N-expr2 s-div2 t-div2 u-div2 v-div2* **by** *simp*

define *sn* **where** *sn = nat s*
define *tn* **where** *tn = nat t*
define *un* **where** *un = nat u*

define *vn* **where** *vn = nat v*
have *seqsn:s^2−s = sn^2 − sn* **using** *stuv sn-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)
have *teqtn:t^2−t = tn^2 − tn* **using** *stuv tn-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)
have *uequn:u^2−u = un^2 − un* **using** *stuv un-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)
have *veqvn:v^2−v = vn^2 − vn* **using** *stuv vn-def*
by (*metis int-nat-eq le-refl of-nat-diff of-nat-power power2-nat-le-imp-le*)

from *N-expr3* **have**
N = (3∗(sn^2−sn) div 2+s)+(3∗(tn^2−tn) div 2+t)+(3∗(un^2−un) div 2+u)+(3∗(vn^2−vn) div 2+ v)+r
using *seqsn teqtn uequn veqvn*
by (*metis* (*mono-tags*, *lifting*) *Num.of-nat-simps*(*5*) *of-nat-numeral zdiv-int*)
hence *N = (3∗(sn^2−sn) div 2+sn)+(3∗(tn^2−tn) div 2+tn)+(3∗(un^2−un) div 2+un)+(3∗(vn^2−vn) div 2+ v)+r*
using *sn-def tn-def un-def stuv int-nat-eq int-ops*(*5*) **by** *presburger*
hence *N = (3∗(sn^2−sn) div 2+sn)+(3∗(tn^2−tn) div 2+tn)+(3∗(un^2−un) div 2+un)+(3∗(vn^2−vn) div 2+ vn)+r*
using *vn-def stuv* **by** (*smt* (*verit*, *del-insts*) *Num.of-nat-simps*(*4*) *int-nat-eq of-nat-eq-iff*)
hence *N = (3∗ sn∗(sn−1) div 2+sn)+(3∗tn∗(tn−1) div 2+tn)+(3∗un∗(un−1) div 2+un)+(3∗ vn∗(vn−1) div 2+ vn)+r*
by (*smt* (*verit*, *ccfv-threshold*) *more-arith-simps*(*11*) *mult.right-neutral power2-eq-square right-diff-distrib′*)
hence *N-expr4:N = polygonal-number 3 sn + polygonal-number 3 tn + polygonal-number 3 un + polygonal-number 3 vn +r*
using *Polygonal-Number-Theorem-Gauss.polygonal-number-def* **by** *presburger*

define *p1* **where** *p1 = polygonal-number 3 sn*
define *p2* **where** *p2 = polygonal-number 3 tn*
define *p3* **where** *p3 = polygonal-number 3 un*
define *p4* **where** *p4 = polygonal-number 3 vn*
have *N-expr5:N = p1 + p2 + p3 + p4 + r* **using** *N-expr4 p1-def p2-def p3-def p4-def* **by** *auto*
thus *?thesis* **using** *p1-def p2-def p3-def p4-def b-r* **by** *blast*
qed
end

2.3 Legendre's Polygonal Number Theorem

We will use the definition of the polygonal numbers from the Gauss Theorem theory file which also imports the Three Squares Theorem AFP entry [1].

theory *Polygonal-Number-Theorem-Legendre*
imports *Polygonal-Number-Theorem-Gauss*
begin

This lemma shows that under certain conditions, an integer N can be written

as the sum of four polygonal numbers.

lemma *sum-of-four-polygonal-numbers*:
fixes $N\ m :: nat$
fixes $b :: int$
assumes $m \geq 3$
assumes $N \geq 2*m$
assumes $[N = b]\ (mod\ m)$
assumes *odd-b*: *odd b*
assumes $b \in \{1/2 + sqrt\ (6*N/m - 3)\ ..\ 2/3 + sqrt\ (8*N/m - 8)\}$
assumes $N \geq 9$
shows $\exists k1\ k2\ k3\ k4.\ N = polygonal\text{-}number\ m\ k1 + polygonal\text{-}number\ m\ k2 + polygonal\text{-}number\ m\ k3 + polygonal\text{-}number\ m\ k4$

proof −
define I **where** $I = \{1/2 + sqrt\ (6*N/m - 3)\ ..\ 2/3 + sqrt\ (8*N/m - 8)\}$
from *assms*(5) *I-def* **have** $b \in I$ **by** *auto*
define a::*int* **where** *a-def*: $a = 2*(N-b)\ div\ m + b$
have $m\ dvd\ (N-b)$ **using** *assms*(3)
by (*smt* (*verit*, *ccfv-threshold*) *cong-iff-dvd-diff zdvd-zdiffD*)
hence $even\ (2*(N-b)\ div\ m)$
by (*metis div-mult-swap dvd-triv-left*)
hence *odd a* **using** *a-def assms*(3) *odd-b* **by** *auto*
from *assms*(1) **have** $m\hat{}3 \geq m$
by (*simp add*: *power3-eq-cube*)
hence $N \geq 2 * m$ **using** *assms*(1,2) **by** *simp*
from *assms*(1) **have** *m-pos*: $m > 0$ **by** *auto*
have $N \geq b$
proof −
from *assms*(1) **have** $m \geq 1$ **by** *auto*
hence $1/m \leq 1$ **using** *m-pos* **by** *auto*
moreover have $N > 0$ **using** ‹$N \geq 2 * m$› *m-pos* **by** *auto*
ultimately have $N/m \leq N$
using *divide-less-eq-1 less-eq-real-def* **by** *fastforce*
hence $sqrt(8*N/m - 8) \leq sqrt(8*(N-1))$ **by** *auto*
from *assms*(1) **have** $m\hat{}3 \geq 3*3*(3::real)$
by (*metis numeral-le-real-of-nat-iff numeral-times-numeral power3-eq-cube power-mono zero-le-numeral*)
from ‹$N \geq 9$› **have** $N-1 \geq 8$ **by** *auto*
hence $(N-1)\hat{}2 \geq 8*(N-1)$ **using** ‹$N > 0$› **by** (*simp add*: *power2-eq-square*)
hence $(N-1) \geq sqrt\ (8*(N-1))$ **using** ‹$N > 0$›
by (*metis of-nat-0-le-iff of-nat-mono of-nat-power real-sqrt-le-mono real-sqrt-pow2 real-sqrt-power*)
hence $N - (1::real) - sqrt(8*N/m - 8) \geq 0$
using ‹$sqrt\ (real\ (8 * N) / real\ m - 8) \leq sqrt\ (real\ (8 * (N - 1)))$› ‹$9 \leq N$› **by** *linarith*
hence *expr-pos*: $N - (2/3::real) - sqrt(8*N/m - 8) \geq 0$ **by** *auto*
have $b \leq 2/3 + sqrt(8*N/m - 8)$ **using** ‹$b \in I$› *I-def* **by** *auto*
hence $N - b \geq N - (2/3 + sqrt(8*N/m-8))$ **by** *auto*
hence $N - b \geq 0$

using *expr-pos of-int-0-le-iff* **by** *auto*
thus *?thesis* **by** *auto*
qed
from ‹$N \geq 2 * m$› *m-pos* **have** $6*N/m - 3 \geq 0$ **by** (*simp add*: *mult-imp-le-div-pos*)
hence $1/2 + sqrt\ (6*N/m - 3) > 0$
by (*smt* (*verit*, *del-insts*) *divide-le-0-1-iff real-sqrt-ge-zero*)
with ‹$b \in I$› *assms*(3) *I-def* **have** $b \geq 1$ **by** *auto*
hence *b-pos*: $b \geq 0$ **by** *auto*
from ‹$b \in I$› **have** *b-in-I*: $(1/2::real) + sqrt\ (6* real\ N\ /\ real\ m - 3) \leq b \wedge b \leq (2/3::real) + sqrt\ (8 * real\ N/real\ m - 8)$ **unfolding** *I-def* **by** *auto*
from *b-pos* ‹$N \geq b$› *a-def* **have** *a-pos*: $a \geq 0$
by (*smt* (*verit*) *m-pos of-nat-0-less-iff pos-imp-zdiv-neg-iff*)
hence $a \geq 1$
by (*smt* (*verit*) ‹*odd a*› *dvd-0-right*)
have $a - b = 2*(N-b)\ div\ m$ **using** *a-def* **by** *auto*
from ‹*int m dvd* (*int N* − *b*)› **have** *m dvd* $2*(N-b)$ **by** *fastforce*
have $a = 2*(N-b)/m + b$ **using** *a-def m-pos*
using ‹*int m dvd 2* ∗ (*int N* − *b*)› **by** *fastforce*
hence $a = 2*N/m - 2*b/m + b$
by (*simp add*: *assms diff-divide-distrib of-nat-diff*)
hence $(2::real)*N/m = a + 2*b/m - b$ **by** *auto*
hence $(2::real)*N = (a + 2*b/m - b)*m$
using *m-pos* **by** (*simp add*: *divide-eq-eq*)
hence $(2::real)*N = m*(a-b) + 2*b$
using ‹*int m dvd 2* ∗ (*int N* − *b*)› *a-def* **by** *auto*
hence $N = m*(a-b)/2 + b$ **by** *auto*
hence *N-expr*: *real N* = *real m* ∗ (*of-int a* − *of-int b*) / 2 + *of-int b* **by** *auto*
have *even* $(a-b)$ **using** ‹*odd a*› ‹*odd b*› **by** *auto*
hence *2 dvd* $m*(a-b)$ **by** *auto*
hence *N-expr2*: $N = m*(a-b)\ div\ 2 + b$ **using** ‹$N = m*(a-b)/2 + b$› **by** *linarith*
define *Nr* **where** $Nr = real\text{-}of\text{-}nat\ N$
define *mr* **where** $mr = real\ m$
define *ar* **where** $ar = real\text{-}of\text{-}int\ a$
define *br* **where** $br = real\text{-}of\text{-}int\ b$
from *assms*(1) **have** $mr \geq 3$ **using** *mr-def* **by** *auto*
moreover have $Nr \geq 2*mr$ **using** *Nr-def mr-def* ‹$N \geq 2 * m$› **by** *auto*
moreover have $br \geq 0$ **using** *br-def b-pos* **by** *auto*
moreover have $mr > 0$ **using** *mr-def m-pos* **by** *auto*
moreover have $ar \geq 0$ **using** *ar-def* ‹$a \geq 0$› **by** *auto*
moreover have $Nr = mr*(ar-br)/2 + br$ **using** *Nr-def mr-def ar-def br-def N-expr* **by** *auto*
moreover have $1/2 + sqrt\ (6*Nr/mr-3) \leq br \wedge br \leq 2/3 + sqrt\ (8*Nr/mr-8)$ **using** *Nr-def mr-def br-def b-in-I* **by** *auto*
ultimately have $br\hat{}2 < 4*ar \wedge 3*ar < br\hat{}2+2*br+4$ **using** *Cauchy-lemma-r-eq-zero*
by *auto*
hence *real-ineq*:$(real\text{-}of\text{-}int\ b)\hat{}2 < 4*(real\text{-}of\text{-}int\ a) \wedge 3*(real\text{-}of\text{-}int\ a) < (real\text{-}of\text{-}int\ b)\hat{}2 + 2*(real\text{-}of\text{-}int\ b) + 4$
using *br-def ar-def* **by** *auto*

hence *int-ineq1*: *b^2<4∗a* **using** *of-int-less-iff* **by** *fastforce*
from *real-ineq* **have** *int-ineq2*: *3∗a < b^2+2∗b+4* **using** *of-int-less-iff* **by** *fastforce*

define *an*:: *nat* **where** *an = nat a*
from *a-pos* **have** *an = a* **unfolding** *an-def* **by** *auto*
define *bn*:: *nat* **where** *bn = nat b*
from *b-pos* **have** *bn = b* **unfolding** *bn-def* **by** *auto*
have *an ≥ 1* **using** ‹*int an = a*› ‹*a ≥ 1*› **by** *auto*
moreover have *bn ≥ 1* **using** ‹*int bn = b*› ‹*b ≥ 1*› **by** *auto*
moreover have *odd an* **using** ‹*odd a*› ‹*int an = a*› **by** *auto*
moreover have *odd bn* **using** ‹*odd b*› ‹*int bn = b*› **by** *auto*
moreover have *bn ^ 2 < 4 ∗ an* **using** *int-ineq1* ‹*int an = a*› ‹*int bn = b*›
using *of-nat-less-iff* **by** *fastforce*
moreover have *3 ∗ an < bn ^ 2 + 2 ∗ bn + 4* **using** *int-ineq2* ‹*int an = a*› ‹*int bn = b*›
using *of-nat-less-iff* **by** *fastforce*
ultimately have ∃ *s t u v* :: *int*. *s ≥ 0 ∧ t ≥ 0 ∧ u ≥ 0 ∧ v ≥ 0 ∧ an = s^2 + t^2 + u^2 + v^2 ∧*
bn = s+t+u+v **using** *four-nonneg-int-sum* **by** *auto*
hence ∃ *s t u v* :: *int*. *s ≥ 0 ∧ t ≥ 0 ∧ u ≥ 0 ∧ v ≥ 0 ∧ a = s^2 + t^2 + u^2 + v^2 ∧*
b = s+t+u+v **using** ‹*int an = a*› ‹*int bn = b*› **by** *auto*
then obtain *s t u v* :: *int* **where** *stuv*: *s ≥ 0 ∧ t ≥ 0 ∧ u ≥ 0 ∧ v ≥ 0 ∧ a = s^2 + t^2 + u^2 + v^2 ∧*
b = s+t+u+v **by** *auto*
hence *N = (m∗(s^2+t^2+u^2+v^2−s−t−u−v) div 2) + s+t+u+v* **using** *N-expr2* **by** (*smt* (*verit*, *ccfv-threshold*))
hence *N = (m∗(s^2−s+t^2−t+u^2−u+v^2−v) div 2) + s+t+u+v* **by** (*smt* (*verit*, *ccfv-SIG*))
hence *N = (m ∗ (s ∗ (s−1) + t ∗ (t−1) + u ∗ (u−1) + v ∗ (v−1)) div 2) +s+t+u+v* **by** (*simp add*: *power2-eq-square algebra-simps*)
hence *previous-step*: *N = (m ∗ s ∗ (s−1) + m ∗ t ∗ (t−1) + m ∗ u ∗ (u−1) + m ∗ v ∗ (v−1)) div 2 + s+t+u+v* **by** (*simp add*: *algebra-simps*)
moreover have *2 dvd m ∗ s ∗ (s−1)* **by** *simp*
moreover have *2 dvd m ∗ t ∗ (t−1)* **by** *simp*
moreover have *2 dvd m ∗ u ∗ (u−1)* **by** *simp*
moreover have *2 dvd m ∗ v ∗ (v−1)* **by** *simp*
ultimately have *N = m ∗ s ∗ (s−1) div 2 + m ∗ t ∗ (t−1) div 2 + m ∗ u ∗ (u−1) div 2 + m ∗ v ∗(v−1) div 2 + s+t+u+v* **by** *fastforce*
hence *N-expr3*: *N = m ∗ s ∗ (s−1) div 2 + s + m ∗ t ∗ (t−1) div 2 + t + m ∗ u ∗ (u−1) div 2 + u + m ∗ v ∗ (v−1) div 2 + v* **by** *auto*
define *sn*::*nat* **where** *sn = nat s*
define *tn*::*nat* **where** *tn = nat t*
define *un*::*nat* **where** *un = nat u*
define *vn*::*nat* **where** *vn = nat v*
have *sn = s* **using** *stuv sn-def* **by** *auto*
hence *m ∗ sn ∗ (sn−1) = m ∗ s ∗ (s−1)* **by** *fastforce*
hence *m ∗ sn ∗ (sn−1) div 2 = m ∗ s ∗ (s−1) div 2* **by** *linarith*

have *tn* = *t* **using** *stuv tn-def* **by** *auto*
hence *m* ∗ *tn* ∗ (*tn*−*1*) = *m* ∗ *t* ∗ (*t*−*1*) **by** *fastforce*
hence *m* ∗ *tn* ∗ (*tn*−*1*) *div 2* = *m* ∗ *t* ∗ (*t*−*1*) *div 2* **by** *linarith*
have *un* = *u* **using** *stuv un-def* **by** *auto*
hence *m* ∗ *un* ∗ (*un*−*1*) = *m* ∗ *u* ∗ (*u*−*1*) **by** *fastforce*
hence *m* ∗ *un* ∗ (*un*−*1*) *div 2* = *m* ∗ *u* ∗ (*u*−*1*) *div 2* **by** *linarith*
have *vn* = *v* **using** *stuv vn-def* **by** *auto*
hence *m* ∗ *vn* ∗ (*vn*−*1*) = *m* ∗ *v* ∗ (*v*−*1*) **by** *fastforce*
hence *m* ∗ *vn* ∗ (*vn*−*1*) *div 2* = *m* ∗ *v* ∗ (*v*−*1*) *div 2* **by** *linarith*
have *N* = *m* ∗ *sn* ∗ (*sn*−*1*) *div 2* + *sn* + *m* ∗ *tn* ∗ (*tn*−*1*) *div 2* + *tn* + *m* ∗ *un* ∗ (*un*−*1*) *div 2* + *un* + *m* ∗ *vn* ∗ (*vn*−*1*) *div 2* + *vn*
using *N-expr3* ‹*sn* = *s*› ‹*tn* = *t*› ‹*un* = *u*› ‹*vn* = *v*› ‹*m* ∗ *sn* ∗ (*sn*−*1*) *div 2* = *m* ∗ *s* ∗ (*s*−*1*) *div 2*› ‹*m* ∗ *tn* ∗ (*tn*−*1*) *div 2* = *m* ∗ *t* ∗ (*t*−*1*) *div 2*› ‹*m* ∗ *un* ∗ (*un*−*1*) *div 2* = *m* ∗ *u* ∗ (*u*−*1*) *div 2*› ‹*m* ∗ *vn* ∗ (*vn*−*1*) *div 2* = *m* ∗ *v* ∗ (*v*−*1*) *div 2*› **by** *linarith*
hence *N* = *polygonal-number m sn* + *polygonal-number m tn* + *polygonal-number m un* + *polygonal-number m vn*
using *Polygonal-Number-Theorem-Gauss.polygonal-number-def* **by** *presburger*
thus *?thesis* **by** *blast*
qed

We show Legendre's polygonal number theorem which corresponds to Theorem 1.10 in [2].

theorem *Legendre-Polygonal-Number-Theorem*:
fixes *m N* :: *nat*
assumes *m* ≥ *3*
assumes *N* ≥ *28∗m^3*
shows *odd m* ⟹ ∃ *k1 k2 k3 k4*::*nat*. *N* = *polygonal-number m k1* + *polygonal-number m k2* + *polygonal-number m k3* + *polygonal-number m k4*
and *even m* ⟹ ∃ *k1 k2 k3 k4 k5*::*nat*. *N* = *polygonal-number m k1* + *polygonal-number m k2* + *polygonal-number m k3* + *polygonal-number m k4* + *polygonal-number m k5* ∧ (*k1* = *0* ∨ *k1* = *1* ∨ *k2* = *0* ∨ *k2* = *1* ∨ *k3* = *0* ∨ *k3* = *1* ∨ *k4* = *0* ∨ *k4* = *1* ∨ *k5* = *0* ∨ *k5* = *1*)

proof −
define *L* :: *real* **where** *L* = (*2/3* + *sqrt* (*8∗N/m* − *8*)) − (*1/2* + *sqrt* (*6∗N/m* − *3*))
define *I* **where** *I* = {*1/2* + *sqrt* (*6∗N/m* − *3*) .. *2/3* + *sqrt* (*8∗N/m* − *8*)}
from *assms*(*1*) **have** *m^3* ≥ *m*
by (*simp add*: *power3-eq-cube*)
hence *N* ≥ *2* ∗ *m* **using** *assms* **by** *simp*
have *m-pos*: *m* > *0* **using** *assms*(*1*) **by** *simp*
have *L* > *2* ∗ *of-nat m* **using** *assms* ‹*N* ≥ *2* ∗ *m*› *m-pos L-def*
apply −
apply (*rule interval-length-greater-than-2m*[**where** *N*=*of-nat N* **and** *m*=*of-nat m*])
apply (*simp-all*)
by (*metis* (*no-types*, *opaque-lifting*) *of-nat-le-iff of-nat-mult of-nat-numeral power3-eq-cube*)

hence *2*: $L > 2 * m$ **by** *simp*
show *thm-odd-m*: $odd\ m \Longrightarrow \exists k1\ k2\ k3\ k4.\ N = polygonal\text{-}number\ m\ k1 + polygonal\text{-}number\ m\ k2 + polygonal\text{-}number\ m\ k3 + polygonal\text{-}number\ m\ k4$
proof −
assume *odd-m*: $odd\ m$
from *assms(1)* **have** $m > 0$ **by** *auto*
define *ce* **where** $ce = \lceil 1/2 + sqrt\ (6*N/m - 3) \rceil$
have $\forall\ i \in \{0..2*m-1\}.\ ce + i \geq ce$ **by** *auto*
hence *lower-bound*: $\forall\ i \in \{0..2*m-1\}.\ ce + i \geq 1/2 + sqrt\ (6*N/m - 3)$ **using** *ceiling-le-iff ce-def* **by** *blast*
have $2*m-1 + ce \leq 2/3 + sqrt\ (8*N/m - 8)$ **using** *2 L-def assms(1) ce-def* **by** *linarith*
hence *upper-bound*: $\forall\ i \in \{0..2*m-1\}.\ ce + i \leq 2/3 + sqrt\ (8*N/m - 8)$ **by** *auto*
from *lower-bound upper-bound* **have** *in-interval*: $\forall\ i \in \{0..2*m-1\}.\ ce + i \in I$ **unfolding** *ce-def I-def* **by** *auto*
have $\exists\ f{::}nat \Rightarrow int.\ (\forall\ i \in \{0..m-1\}.\ odd\ (f\ i)) \wedge (\forall\ i \in \{1..m-1\}.\ f\ i = f\ 0 + 2*i) \wedge (\forall\ i \in \{0..m-1\}.\ f\ i \in I)$
proof −
have *?thesis* **if** *odd-f0*: $odd\ ce$
proof −
define $g{::}nat \Rightarrow int$ **where** $g\ i = ce + 2*i$
have $odd\ (g\ 0)$ **using** *odd-f0* ‹$g \equiv \lambda i.\ ce + int\ (2 * i)$› **by** *auto*
hence $\forall\ i \in \{0..m-1\}.\ odd\ (g\ i)$ **using** ‹$g \equiv \lambda i.\ ce + int\ (2 * i)$› **by** *auto*
have $\forall\ i \in \{1..m-1\}.\ g\ i = g\ 0 + 2*i$ **using** ‹$g \equiv \lambda i.\ ce + int\ (2 * i)$› **by** *auto*
have $\forall\ i \in \{0..m-1\}.\ 2*i < 2*m-1$ **using** *m-pos* **by** *auto*
hence $\forall\ i \in \{0..m-1\}.\ g\ i \in I$ **using** ‹$g \equiv \lambda i.\ ce + int\ (2 * i)$› *in-interval* **by** *fastforce*
show *?thesis* **using** ‹$\forall i \in \{0..m-1\}.\ odd\ (g\ i)$› ‹$\forall i \in \{0..m-1\}.\ real\text{-}of\text{-}int\ (g\ i) \in I$› ‹$\forall i \in \{1..m-1\}.\ g\ i = g\ 0 + int(2 * i)$› **by** *blast*
qed
moreover have *?thesis* **if** *even ce*
proof −
from ‹*even ce*› **have** *odd-f1*: $odd\ (ce + 1)$ **by** *auto*
define $g{::}nat \Rightarrow int$ **where** $g\ i = ce + (2*i + 1)$
have $odd\ (g\ 0)$ **using** *odd-f1* ‹$g \equiv \lambda i.\ ce + int\ (2 * i + 1)$› **by** *auto*
hence $\forall\ i \in \{0..m-1\}.\ odd\ (g\ i)$ **using** ‹$g \equiv \lambda i.\ ce + int\ (2 * i + 1)$› **by** *auto*
have $\forall\ i \in \{1..m-1\}.\ g\ i = g\ 0 + 2*i$ **using** ‹$g \equiv \lambda i.\ ce + int\ (2 * i + 1)$› **by** *auto*
have $\forall\ i \in \{0..m-1\}.\ 2*i + 1 \leq 2*m-1$ **using** *m-pos* **by** *auto*
hence $\forall\ i \in \{0..m-1\}.\ g\ i \in I$ **using** ‹$g \equiv \lambda i.\ ce + int\ (2 * i + 1)$› *in-interval* **by** *fastforce*
show *?thesis* **using** ‹$\forall i \in \{0..m-1\}.\ odd\ (g\ i)$› ‹$\forall i \in \{0..m-1\}.\ real\text{-}of\text{-}int\ (g\ i) \in I$› ‹$\forall i \in \{1..m-1\}.\ g\ i = g\ 0 + int(2 * i)$› **by** *blast*
qed
ultimately show *?thesis* **by** *blast*
qed

then obtain f::$nat \Rightarrow int$ **where** *f-def*: $(\forall\ i{\in}\{0..m{-}1\}.\ odd\ (f\ i)) \wedge (\forall\ i{\in}\{1..m{-}1\}.\ f\ i = f\ 0 + 2{*}i) \wedge (\forall\ i{\in}\{0..m{-}1\}.\ f\ i \in I)$ **by** *auto*

have *inj-lemma*: $[\![i \in \{0..m{-}1\};\ j \in \{0..m{-}1\};\ [f\ i = f\ j]\ (mod\ m)]\!] \Longrightarrow i = j$ **for** $i\ j$
proof −
assume *asm1*: $i \in \{0..m{-}1\}$
assume *asm2*: $j \in \{0..m{-}1\}$
assume *asm3*: $[f\ i = f\ j]\ (mod\ m)$
from *f-def* **have** $odd\ (f\ 0)$ **by** *auto*
hence $\exists\ k$::$int.\ f\ 0 = 2{*}k + 1$ **by** (*metis oddE*)
then obtain k::int **where** *k-def*: $f\ 0 = 2{*}k + 1$ **by** *auto*
have *False* **if** *case2*: $i = 0 \wedge j > 0$
proof −
have $f\ j = f\ 0 + 2{*}j$ **using** *f-def case2 asm2* **by** *auto*
hence $[2{*}k + 1 = 2{*}k + 1 + 2{*}j]\ (mod\ m)$ **using** *asm3 case2 k-def* **by** *auto*
hence $[2{*}j = 0]\ (mod\ m)$
by (*metis cong-add-lcancel-0 cong-int-iff cong-sym-eq int-ops*(1))
have *coprime 2 m* **using** *odd-m* **by** *simp*
hence $[j = 0]\ (mod\ m)$ **using** ‹$[2{*}j = 0]\ (mod\ m)$› **by** (*simp add*: *cong-0-iff coprime-dvd-mult-right-iff*)
thus *False* **using** *asm2 case2 cong-less-modulus-unique-nat* **by** *fastforce*
qed
moreover have *False* **if** *case3*: $i > 0 \wedge j = 0$
proof −
have $f\ i = f\ 0 + 2{*}i$ **using** *f-def case3 asm1* **by** *auto*
hence $[2{*}k + 1 + 2{*}i = 2{*}k + 1]\ (mod\ m)$ **using** *asm3 case3 k-def* **by** *auto*
hence $[2{*}i = 0]\ (mod\ m)$
by (*metis cong-add-lcancel-0 cong-int-iff cong-sym-eq int-ops*(1))
have *coprime 2 m* **using** *odd-m* **by** *simp*
hence $[i = 0]\ (mod\ m)$ **using** ‹$[2{*}i = 0]\ (mod\ m)$› **by** (*simp add*: *cong-0-iff coprime-dvd-mult-right-iff*)
thus *False* **using** *asm1 case3 cong-less-modulus-unique-nat* **by** *fastforce*
qed
moreover have *?thesis* **if** *case4*: $i > 0 \wedge j > 0$
proof −
have $i > 0$ **and** $j > 0$ **using** *case4* **by** *auto*
have $f\ i = f\ 0 + 2{*}i$ **using** *f-def case4 asm1* **by** *auto*
moreover have $f\ j = f\ 0 + 2{*}j$ **using** *f-def case4 asm2* **by** *auto*
ultimately have $[2{*}k + 1 + 2{*}i = 2{*}k + 1 + 2{*}j]\ (mod\ m)$ **using** *case4 k-def asm3* **by** *fastforce*
hence $[2{*}i = 2{*}j]\ (mod\ m)$
using *cong-add-lcancel cong-int-iff* **by** *blast*
have *coprime 2 m* **using** *odd-m* **by** *simp*
hence $[i = j]\ (mod\ m)$
using ‹$[2 * i = 2 * j]\ (mod\ m)$› *cong-mult-lcancel-nat* **by** *auto*
thus *?thesis* **using** *asm1 asm2 case4 cong-less-modulus-unique-nat* **by** *auto*

qed
ultimately show *?thesis* **by** *fastforce*
qed
have *complete-cong-class*: $\exists\, i \in \{0..m-1\}.\ [f\ i = S]\ (mod\ m)$ **for** S
proof −
have $(f\ i)\ mod\ m = (f\ j)\ mod\ m \Longrightarrow [f\ i = f\ j]\ (mod\ m)$ **for** $i\ j$
by (*simp add*: *unique-euclidean-semiring-class.cong-def*)
hence *inj2*: $[\![i \in \{0..m-1\};\ j \in \{0..m-1\};\ (f\ i)\ mod\ m = (f\ j)\ mod\ m]\!] \Longrightarrow i = j$ **for** $i\ j$
using *inj-lemma* **by** *auto*
hence *injective*: $\forall\, i \in \{0..m-1\}.\ \forall\, j \in \{0..m-1\}.\ (f\ i)\ mod\ m = (f\ j)\ mod\ m \longrightarrow i = j$
by *auto*
define $g :: nat \Rightarrow int$ **where** $g\ i = (f\ i)\ mod\ m$
then have *g-inj2*: $\forall\, i \in \{0..m-1\}.\ \forall\ j \in \{0..m-1\}.\ g\ i = g\ j \longrightarrow i = j$
using ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$› *injective* **by** *fastforce*
then have *g-inj*: *inj-on g* $\{0..m-1\}$
by (*meson inj-onI*)
have *g-range2*: $\forall\ i \in \{0..m-1\}.\ g\ i \in \{0..m-1\}$ **using** ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$›
by (*metis m-pos Euclidean-Rings.pos-mod-bound Euclidean-Rings.pos-mod-sign atLeastAtMost-iff mod-by-1 mod-if not-gr0 of-nat-0-less-iff of-nat-1 of-nat-diff verit-comp-simplify1*(*3*) *zle-diff1-eq*)
hence *image-subset*: $g\ `\ \{0..m-1\} \subseteq \{0..m-1\}$ **by** *blast*
have *g-range*: $i \in \{0..m-1\} \Longrightarrow g\ i \in \{0..m-1\}$ **using** ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$›
by (*metis m-pos Euclidean-Rings.pos-mod-bound Euclidean-Rings.pos-mod-sign atLeastAtMost-iff mod-by-1 mod-if not-gr0 of-nat-0-less-iff of-nat-1 of-nat-diff verit-comp-simplify1*(*3*) *zle-diff1-eq*)
have *card-ge-m*: $card\ (g\ `\ \{0..m-1\}) \geq m$ **using** *g-inj*
by (*metis m-pos Suc-diff-1 card-atLeastAtMost card-image minus-nat.diff-0 verit-comp-simplify1*(*2*))
have $card\ \{0..m-1\} = m$ **using** *m-pos* **by** *force*
hence *card-le-m*: $card\ (g\ `\ \{0..m-1\}) \leq m$ **using** *m-pos*
by (*metis card-image g-inj le-refl*)
from *card-ge-m card-le-m* **have** *image-size*: $card\ (g\ `\ \{0..m-1\}) = m$ **by** *auto*
with ‹$card\ \{0..m-1\} = m$› **have** *equal-card*: $card\ (g\ `\ \{0..m-1\}) = card\ \{0..m-1\}$ **by** *auto*
have *finite* $(g\ `\ \{0..m-1\})$ **using** *image-size* **by** *auto*
with *equal-card image-subset* **have** $g\ `\ \{0..m-1\} = \{0..m-1\}$
by (*metis card-image card-subset-eq finite-atLeastAtMost-int image-int-atLeastAtMost inj-on-of-nat of-nat-0*)
hence $i \in \{0..m-1\} \Longrightarrow \exists\ j \in \{0..m-1\}.\ i = g\ j$ **for** i **by** *auto*
hence $i \in \{0..m-1\} \Longrightarrow \exists\ j \in \{0..m-1\}.\ i = (f\ j)\ mod\ m$ **for** i
using ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$› **by** *auto*
hence *surj*: $i \in \{0..m-1\} \Longrightarrow \exists\ j \in \{0..m-1\}.\ [i = f\ j]\ (mod\ m)$ **for** i
by (*metis mod-mod-trivial unique-euclidean-semiring-class.cong-def*)
have $S\ mod\ m \geq 0$ **using** *m-pos* **by** *simp*
moreover have $S\ mod\ m \leq m-1$

using *m-pos* **by** (*simp add*: *of-nat-diff*)
ultimately have $S\ mod\ m \in \{0..m-1\}$ **by** *auto*
with *surj m-pos* **have** $\exists\ j \in \{0..m-1\}.\ [S\ mod\ m = f\ j]\ (mod\ m)$
by (*metis atLeastAtMost-iff less-eq-nat.simps*(1) *nonneg-int-cases of-nat-less-iff verit-comp-simplify*(3))
thus *?thesis* **using** *cong-mod-right cong-sym* **by** *blast*
qed
have $\exists\ b{::}int.\ [N = b]\ (mod\ m) \wedge odd\ b \wedge b \in I$
proof −
have $N\ mod\ m \geq 0$ **by** *auto*
moreover have $N\ mod\ m \leq m-1$
using *m-pos less-Suc-eq-le* **by** *fastforce*
ultimately have $N\ mod\ m \in \{0..m-1\}$ **by** *auto*
with *complete-cong-class* **have** $\exists\ i.\ i \in \{0..m-1\} \wedge [f\ i = N]\ (mod\ m)$ **by** *blast*
then obtain $c{::}nat$ **where** *c-def*: $c \in \{0..m-1\} \wedge [f\ c = N]\ (mod\ m)$ **by** *auto*
define $b{::}int$ **where** $b = f\ c$
have $[N = b]\ (mod\ m)$ **using** *b-def c-def* **by** (*metis cong-sym*)
moreover have *odd b* **using** *b-def f-def c-def* **by** *auto*
moreover have $b \in I$ **using** *b-def f-def c-def* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
then obtain $b{::}int$ **where** *b-def*: $[N = b]\ (mod\ m) \wedge odd\ b \wedge b \in I$ **by** *auto*
have $m\hat{}3 \geq m$ **using** *m-pos* **by** (*auto simp add*: *power3-eq-cube*)
hence $N \geq 28*m$ **using** *assms*(1,2) **by** *linarith*
hence $N \geq 2*m$ **by** *simp*
have $m\hat{}3 \geq 3*3*(3{::}nat)$ **using** *assms*(1)
by (*metis power3-eq-cube power-mono zero-le-numeral*)
hence $N \geq 28*3*3*(3{::}nat)$ **using** *assms*(2) **by** *auto*
hence $N \geq 9$ **by** *simp*
show *?thesis* **using** *sum-of-four-polygonal-numbers assms*(1) *b-def I-def* ‹$N \geq 2 * m$› ‹$N \geq 9$› **by** *blast*
qed
show *thm-even-m*: $even\ m \Longrightarrow \exists\ k1\ k2\ k3\ k4\ k5.\ N = polygonal\text{-}number\ m\ k1 + polygonal\text{-}number\ m\ k2 + polygonal\text{-}number\ m\ k3 + polygonal\text{-}number\ m\ k4 + polygonal\text{-}number\ m\ k5 \wedge (k1 = 0 \vee k1 = 1 \vee k2 = 0 \vee k2 = 1 \vee k3 = 0 \vee k3 = 1 \vee k4 = 0 \vee k4 = 1 \vee k5 = 0 \vee k5 = 1)$
proof −
assume *even-m*: *even m*
from *assms*(1) **have** $m > 0$ **by** *auto*
define *ce* **where** $ce = \lceil 1/2 + sqrt\ (6*N/m - 3) \rceil$
have $\forall\ i{\in}\{0..m-1\}.\ ce + i \geq ce$ **by** *auto*
hence *lower-bound*: $\forall\ i{\in}\{0..m-1\}.\ ce + i \geq 1/2 + sqrt\ (6*N/m - 3)$ **using** *ceiling-le-iff ce-def* **by** *blast*
have $m-1 + ce \leq 2/3 + sqrt\ (8*N/m - 8)$ **using** *2 L-def assms*(1) *ce-def* **by** *linarith*
hence *upper-bound*: $\forall\ i{\in}\{0..m-1\}.\ ce + i \leq\ 2/3 + sqrt\ (8*N/m - 8)$ **by** *auto*

from *lower-bound upper-bound* **have** *in-interval*: $\forall\ i{\in}\{0..m{-}1\}.\ ce + i \in I$ **unfolding** *ce-def I-def* **by** *auto*
have $\exists\ f{::}nat \Rightarrow int.\ (\forall\ i{\in}\{1..m{-}1\}.\ f\ i = f\ 0 + i) \wedge (\forall\ i{\in}\{0..m{-}1\}.\ f\ i \in I)$
proof −
define *g*::*nat* ⇒ *int* **where** $g\ i = ce + i$
have $\forall\ i{\in}\{1..m{-}1\}.\ g\ i = g\ 0 + i$ **using** ‹$g \equiv \lambda i.\ ce + int\ i$› **by** *auto*
have $\forall\ i{\in}\{0..m{-}1\}.\ i < m$ **using** *m-pos* **by** *auto*
hence $\forall\ i{\in}\{0..m{-}1\}.\ g\ i \in I$ **using** ‹$g \equiv \lambda i.\ ce + int\ i$› *in-interval* **by** *fastforce*
show *?thesis* **by** (*metis Num.of-nat-simps*(1) ‹$\forall\ i{\in}\{0..m - 1\}.$ *real-of-int* $(g\ i) \in I$› ‹$g \equiv \lambda i.\ ce + int\ i$› *arith-extra-simps*(6))
qed
then obtain *f*::*nat* ⇒ *int* **where** *f-def*: $(\forall\ i{\in}\{1..m{-}1\}.\ f\ i = f\ 0 + i) \wedge (\forall\ i{\in}\{0..m{-}1\}.\ f\ i \in I)$ **by** *auto*
have *inj-lemma*: $[\![i \in \{0..m{-}1\};\ j \in \{0..m{-}1\};\ [f\ i = f\ j]\ (mod\ m)]\!] \Longrightarrow i = j$ **for** *i j*
proof −
assume *asm1*: $i \in \{0..m{-}1\}$
assume *asm2*: $j \in \{0..m{-}1\}$
assume *asm3*: $[f\ i = f\ j]\ (mod\ m)$
have *False* **if** *case2*: $i = 0 \wedge j > 0$
proof −
have $f\ j = f\ 0 + j$ **using** *f-def case2 asm2* **by** *auto*
hence $[f\ 0 = f\ 0 + j]\ (mod\ m)$ **using** *asm3 case2* **by** *auto*
hence $[j = 0]\ (mod\ m)$
by (*metis cong-add-lcancel-0 cong-int-iff cong-sym-eq int-ops*(1))
thus *False* **using** *asm2 case2 cong-less-modulus-unique-nat* **by** *fastforce*
qed
moreover have *False* **if** *case3*: $i > 0 \wedge j = 0$
proof −
have $f\ i = f\ 0 + i$ **using** *f-def case3 asm1* **by** *auto*
hence $[f\ 0 + i = f\ 0]\ (mod\ m)$ **using** *asm3 case3* **by** *auto*
hence $[i = 0]\ (mod\ m)$
by (*metis cong-add-lcancel-0 cong-int-iff cong-sym-eq int-ops*(1))
thus *False* **using** *asm1 case3 cong-less-modulus-unique-nat* **by** *fastforce*
qed
moreover have *?thesis* **if** *case4*: $i > 0 \wedge j > 0$
proof −
have $i > 0$ **and** $j > 0$ **using** *case4* **by** *auto*
have $f\ i = f\ 0 + i$ **using** *f-def case4 asm1* **by** *auto*
moreover have $f\ j = f\ 0 + j$ **using** *f-def case4 asm2* **by** *auto*
ultimately have $[f\ 0 + i = f\ 0 + j]\ (mod\ m)$ **using** *case4 asm3* **by** *fastforce*
hence $[i = j]\ (mod\ m)$
using *cong-add-lcancel cong-int-iff* **by** *blast*
thus *?thesis* **using** *asm1 asm2 case4 cong-less-modulus-unique-nat* **by** *auto*
qed
ultimately show *?thesis* **by** *fastforce*
qed

have *complete-cong-class*: $\exists\, i \in \{0..m-1\}.\ [f\ i = S]\ (mod\ m)$ **for** S
proof −
have $(f\ i)\ mod\ m = (f\ j)\ mod\ m \Longrightarrow [f\ i = f\ j]\ (mod\ m)$ **for** $i\ j$
by (*simp add*: *unique-euclidean-semiring-class.cong-def*)
hence *inj2*: $[\![i \in \{0..m-1\};\ j \in \{0..m-1\};\ (f\ i)\ mod\ m = (f\ j)\ mod\ m]\!] \Longrightarrow i = j$ **for** $i\ j$
using *inj-lemma* **by** *auto*
hence *injective*: $\forall\, i \in \{0..m-1\}.\ \forall\, j \in \{0..m-1\}.\ (f\ i)\ mod\ m = (f\ j)\ mod\ m \longrightarrow i = j$
by *auto*
define $g :: nat \Rightarrow int$ **where** $g\ i = (f\ i)\ mod\ m$
then have *g-inj2*: $\forall\, i \in \{0..m-1\}.\ \forall\ j \in \{0..m-1\}.\ g\ i = g\ j \longrightarrow i = j$
using ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$› *injective* **by** *fastforce*
then have *g-inj*: *inj-on g* $\{0..m-1\}$
by (*meson inj-onI*)
have *g-range2*: $\forall\ i \in \{0..m-1\}.\ g\ i \in \{0..m-1\}$ **using** ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$›
by (*metis m-pos Euclidean-Rings.pos-mod-bound Euclidean-Rings.pos-mod-sign atLeastAtMost-iff mod-by-1 mod-if not-gr0 of-nat-0-less-iff of-nat-1 of-nat-diff verit-comp-simplify1*(*3*) *zle-diff1-eq*)
hence *image-subset*: $g\ `\ \{0..m-1\} \subseteq \{0..m-1\}$ **by** *blast*
have *g-range*: $i \in \{0..m-1\} \Longrightarrow g\ i \in \{0..m-1\}$ **using** ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$›
by (*metis m-pos Euclidean-Rings.pos-mod-bound Euclidean-Rings.pos-mod-sign atLeastAtMost-iff mod-by-1 mod-if not-gr0 of-nat-0-less-iff of-nat-1 of-nat-diff verit-comp-simplify1*(*3*) *zle-diff1-eq*)
have *card-ge-m*: $card\ (g\ `\ \{0..m-1\}) \geq m$ **using** *g-inj*
by (*metis m-pos Suc-diff-1 card-atLeastAtMost card-image minus-nat.diff-0 verit-comp-simplify1*(*2*))
have $card\ \{0..m-1\} = m$ **using** *m-pos* **by** *force*
hence *card-le-m*: $card\ (g\ `\ \{0..m-1\}) \leq m$ **using** *m-pos*
by (*metis card-image g-inj le-refl*)
from *card-ge-m card-le-m* **have** *image-size*: $card\ (g\ `\ \{0..m-1\}) = m$ **by** *auto*
with ‹$card\ \{0..m-1\} = m$› **have** *equal-card*: $card\ (g\ `\ \{0..m-1\}) = card\ \{0..m-1\}$ **by** *auto*
have *finite* $(g\ `\ \{0..m-1\})$ **using** *image-size* **by** *auto*
with *equal-card image-subset* **have** $g\ `\ \{0..m-1\} = \{0..m-1\}$
by (*metis card-image card-subset-eq finite-atLeastAtMost-int image-int-atLeastAtMost inj-on-of-nat of-nat-0*)
hence $i \in \{0..m-1\} \Longrightarrow \exists\ j \in \{0..m-1\}.\ i = g\ j$ **for** i **by** *auto*
hence $i \in \{0..m-1\} \Longrightarrow \exists\ j \in \{0..m-1\}.\ i = (f\ j)\ mod\ m$ **for** i
using ‹$g \equiv \lambda i.\ f\ i\ mod\ int\ m$› **by** *auto*
hence *surj*: $i \in \{0..m-1\} \Longrightarrow \exists\ j \in \{0..m-1\}.\ [i = f\ j]\ (mod\ m)$ **for** i
by (*metis mod-mod-trivial unique-euclidean-semiring-class.cong-def*)
have $S\ mod\ m \geq 0$ **using** *m-pos* **by** *simp*
moreover have $S\ mod\ m \leq m-1$
using *m-pos* **by** (*simp add*: *of-nat-diff*)
ultimately have $S\ mod\ m \in \{0..m-1\}$ **by** *auto*
with *surj m-pos* **have** $\exists\ j \in \{0..m-1\}.\ [S\ mod\ m = f\ j]\ (mod\ m)$

by (*metis atLeastAtMost-iff less-eq-nat.simps(1) nonneg-int-cases of-nat-less-iff verit-comp-simplify(3)*)
thus *?thesis* **using** *cong-mod-right cong-sym* **by** *blast*
qed
have *thm-odd-n*: *?thesis* **if** *odd N*
proof −
have ∃ *b*::*int*. [*N* = *b*] (*mod m*) ∧ *odd b* ∧ *b* ∈ *I*
proof −
from *complete-cong-class* **have** ∃ *i*. *i* ∈ {*0*..*m*−*1*} ∧ [*f i* = *N*] (*mod m*) **by** *blast*
then obtain *c*::*nat* **where** *c-def*: *c* ∈ {*0*..*m*−*1*} ∧ [*f c* = *N*] (*mod m*) **by** *auto*
define *b*::*int* **where** *b* = *f c*
have [*N* = *b*] (*mod m*) **using** *b-def c-def* **by** (*metis cong-sym*)
moreover have *odd b*
proof
assume *even b*
have ∃*k*::*int*. *N* = *b* + *k*∗*m* **using** ‹[*N* = *b*] (*mod m*)›
by (*metis cong-iff-lin cong-sym-eq mult.commute*)
then obtain *k*::*int* **where** *k-def*: *N* = *b* + *k*∗*m* **by** *auto*
have *even* (*k*∗*m*) **using** *even-m* **by** *auto*
hence *even N* **using** *k-def* ‹*even b*› **by** *presburger*
thus *False* **using** ‹*odd N*› **by** *blast*
qed
moreover have *b* ∈ *I* **using** *b-def f-def c-def* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
then obtain *b*::*int* **where** *b-def*: [*N* = *b*] (*mod m*) ∧ *odd b* ∧ *b* ∈ *I* **by** *auto*
have *m*^*3* ≥ *m* **using** *m-pos* **by** (*auto simp add*: *power3-eq-cube*)
hence *N* ≥ *28*∗*m* **using** *assms*(*1*,*2*) **by** *linarith*
hence *N* ≥ *2*∗*m* **by** *simp*
have *m*^*3* ≥ *3*∗*3*∗(*3*::*nat*) **using** *assms*(*1*)
by (*metis power3-eq-cube power-mono zero-le-numeral*)
hence *N* ≥ *28*∗*3*∗*3*∗(*3*::*nat*) **using** *assms*(*2*) **by** *auto*
hence *N* ≥ *9* **by** *simp*
hence ∃*k1 k2 k3 k4*. *N* = *polygonal-number m k1* + *polygonal-number m k2* + *polygonal-number m k3* + *polygonal-number m k4*
using *sum-of-four-polygonal-numbers assms*(*1*) *b-def I-def* ‹*N* ≥ *2* ∗ *m*› ‹*N* ≥ *9*› **by** *blast*
then obtain *k1 k2 k3 k4* **where** *N* = *polygonal-number m k1* + *polygonal-number m k2* + *polygonal-number m k3* + *polygonal-number m k4* **by** *blast*
moreover have *polygonal-number m 0* = *0* **using** *Polygonal-Number-Theorem-Gauss.polygonal-number-de*
by *auto*
ultimately have *N* = *polygonal-number m k1* + *polygonal-number m k2* + *polygonal-number m k3* + *polygonal-number m k4* + *polygonal-number m 0* **by** *linarith*
thus *?thesis* **by** *blast*
qed
have *thm-even-n*: *?thesis* **if** *even N*

```
  proof −
    have ∃ b::int. [N−1 = b] (mod m) ∧ odd b ∧ b ∈ I
    proof −
      from complete-cong-class have ∃ i. i ∈ {0..m−1} ∧ [f i = N−1] (mod m)
by blast
      then obtain c::nat where c-def: c ∈ {0..m−1} ∧ [f c = N−1] (mod m)
by auto
      define b::int where b = f c
      have [N−1 = b] (mod m) using b-def c-def by (metis cong-sym)
      moreover have odd b
      proof
        assume even b
        have ∃k::int. N−1 = b + k∗m using ‹[N−1 = b] (mod m)›
        by (metis (full-types) cong-iff-lin cong-sym-eq mult.commute)
        then obtain k::int where k-def: N−1 = b + k∗m by auto
        have even (k∗m) using even-m by auto
        hence even (N−1) using k-def ‹even b› by presburger
        hence odd N
          by (metis Groups.mult-ac(2) ‹2 ∗ m ≤ N› add-eq-self-zero add-leD1
assms(1) dvd-diffD1 le-trans mult-2-right nat-1-add-1 nat-dvd-1-iff-1 rel-simps(25)
zero-neq-numeral)
        thus False using ‹even N› by blast
      qed
      moreover have b ∈ I using b-def f-def c-def by auto
      ultimately show ?thesis by auto
    qed
    then obtain b::int where b-def: [N−1 = b] (mod m) ∧ odd b ∧ b ∈ I by
auto
    from b-def have b ∈ I by auto
    define a::int where a-def: a = 2∗(N−1−b) div m + b
    have m dvd (N−1−b) using b-def
      by (smt (verit, ccfv-threshold) cong-iff-dvd-diff zdvd-zdiffD)
    hence even (2∗(N−1−b) div m)
      by (metis div-mult-swap dvd-triv-left)
    hence odd a using a-def b-def by auto
    from assms(1) have m^3 ≥ m
      by (simp add: power3-eq-cube)
    hence N ≥ 2 ∗ m using assms(1,2) by simp
    from assms(1) have m-pos: m > 0 by auto
    have N−1 ≥ b
    proof −
      from assms(1) have m ≥ 1 by auto
      hence 1/m ≤ 1 using m-pos by auto
      moreover have N > 0 using ‹N ≥ 2 ∗ m› m-pos by auto
      ultimately have N/m ≤ N
        using divide-less-eq-1 less-eq-real-def by fastforce
      hence sqrt(8∗N/m − 8) ≤ sqrt(8∗(N−1)) by auto
      from assms(1) have m^3 ≥ 3∗3∗(3::real)
        by (metis numeral-le-real-of-nat-iff numeral-times-numeral power3-eq-cube
```

power-mono zero-le-numeral)
hence $N \geq 28*3*3*(3::real)$ **using** *assms*(2) **by** *linarith*
hence $N-6 \geq 6$ **by** *simp*
hence $N-6 \geq 0$ **by** *simp*
with ‹$N-6 \geq 6$› **have** $(N-6)\hat{}2 \geq 6\hat{}2$
using *power2-nat-le-eq-le* **by** *blast*
hence $(N-6)\hat{}2 \geq 24$ **by** *simp*
hence $(N-2)\hat{}2 \geq 8*(N-1)$ **by** (*simp add*: *power2-eq-square algebra-simps*)
hence $(N-2) \geq sqrt\ (8*(N-1))$ **using** ‹$N > 0$›
by (*metis of-nat-0-le-iff of-nat-mono of-nat-power real-sqrt-le-mono real-sqrt-pow2 real-sqrt-power*)
hence $N - (2::real) - sqrt(8*N/m - 8) \geq 0$
using ‹$sqrt\ (real\ (8 * N) / real\ m - 8) \leq sqrt\ (real\ (8 * (N - 1)))$› ‹$N-6 \geq 6$› **by** *linarith*
hence *expr-pos*: $N - 1 - (2/3::real) - sqrt(8*N/m - 8) \geq 0$ **by** *auto*
have $b \leq 2/3 + sqrt(8*N/m - 8)$ **using** ‹$b \in I$› *I-def* **by** *auto*
hence $N - 1 - b \geq N - 1 - (2/3 + sqrt(8*N/m-8))$ **by** *auto*
hence $N - 1 - b \geq 0$
using *expr-pos of-int-0-le-iff* **by** *auto*
thus *?thesis* **by** *auto*
qed
from ‹$N \geq 2 * m$› *m-pos* **have** $6*N/m - 3 \geq 0$ **by** (*simp add*: *mult-imp-le-div-pos*)
hence $1/2 + sqrt\ (6*N/m - 3) > 0$
by (*smt* (*verit*, *del-insts*) *divide-le-0-1-iff real-sqrt-ge-zero*)
with ‹$b \in I$› *b-def I-def* **have** $b \geq 1$ **by** *auto*
hence *b-pos*: $b \geq 0$ **by** *auto*
from ‹$b \in I$› **have** *b-in-I*: $(1/2::real) + sqrt\ (6* real\ N / real\ m - 3) \leq b \wedge b \leq (2/3::real) + sqrt\ (8 * real\ N/real\ m - 8)$ **unfolding** *I-def* **by** *auto*
from *b-pos* ‹$N-1 \geq b$› *a-def* **have** *a-pos*: $a \geq 0$
by (*smt* (*verit*) *m-pos of-nat-0-less-iff pos-imp-zdiv-neg-iff*)
hence $a \geq 1$
by (*smt* (*verit*) ‹*odd a*› *dvd-0-right*)
have $a - b = 2*(N-1-b)\ div\ m$ **using** *a-def* **by** *auto*
from ‹*int m dvd* $(N - 1 - b)$› **have** $m\ dvd\ 2*(N-1-b)$ **by** *fastforce*
have $a = 2*(N-1-b)/m + b$ **using** *a-def m-pos*
using ‹*int m dvd* $2 * (N - 1 - b)$› **by** *fastforce*
hence $a = 2*(N-1)/m - 2*b/m + b$
by (*simp add*: *assms diff-divide-distrib of-nat-diff*)
hence $(2::real)*(N-1)/m = a + 2*b/m - b$ **by** *auto*
hence $(2::real)*(N-1) = (a + 2*b/m - b)*m$
using *m-pos* **by** (*simp add*: *divide-eq-eq*)
hence $(2::real)*(N-1) = m*(a-b) + 2*b$
using ‹*int m dvd* $2 * (N - 1 - b)$› *a-def* **by** *auto*
hence $N-1 = m*(a-b)/2 + b$ **by** *auto*
hence $N = m*(a-b)/2 + b + 1$
using ‹$2 * m \leq N$› *assms*(1) **by** *linarith*
hence *N-expr*: $N = real\ m * (of\text{-}int\ a - of\text{-}int\ b) / 2 + of\text{-}int\ b + 1$ **by** *auto*
have *even* $(a-b)$ **using** ‹*odd a*› *b-def* **by** *auto*
hence *2 dvd* $m*(a-b)$ **by** *auto*

hence *N-expr2*: $N = m*(a-b)$ *div* $2 + b + 1$ **using** ‹$N = m*(a-b)/2 + b + 1$› **by** *linarith*
define *Nr* **where** $Nr = real\text{-}of\text{-}nat\ N$
define *mr* **where** $mr = real\ m$
define *ar* **where** $ar = real\text{-}of\text{-}int\ a$
define *br* **where** $br = real\text{-}of\text{-}int\ b$
from *assms*(*1*) **have** $mr \geq 3$ **using** *mr-def* **by** *auto*
moreover have $Nr \geq 2*mr$ **using** *Nr-def mr-def* ‹$N \geq 2 * m$› **by** *auto*
moreover have $br \geq 0$ **using** *br-def b-pos* **by** *auto*
moreover have $mr > 0$ **using** *mr-def m-pos* **by** *auto*
moreover have $ar \geq 0$ **using** *ar-def* ‹$a \geq 0$› **by** *auto*
moreover have $Nr = mr*(ar-br)/2 + br + 1$ **using** *Nr-def mr-def ar-def br-def N-expr* **by** *auto*
moreover have $1/2 + sqrt\ (6*Nr/mr-3) \leq br \wedge br \leq 2/3 + sqrt\ (8*Nr/mr-8)$ **using** *Nr-def mr-def br-def b-in-I* **by** *auto*
ultimately have $br\hat{}2 < 4*ar \wedge 3*ar < br\hat{}2+2*br+4$ **using** *Cauchy-lemma* **by** *auto*
hence *real-ineq*:$(real\text{-}of\text{-}int\ b)\hat{}2 < 4*(real\text{-}of\text{-}int\ a) \wedge 3*(real\text{-}of\text{-}int\ a) < (real\text{-}of\text{-}int\ b)\hat{}2 + 2*(real\text{-}of\text{-}int\ b) + 4$
using *br-def ar-def* **by** *auto*
hence *int-ineq1*: $b\hat{}2<4*a$ **using** *of-int-less-iff* **by** *fastforce*
from *real-ineq* **have** *int-ineq2*: $3*a < b\hat{}2+2*b+4$ **using** *of-int-less-iff* **by** *fastforce*

define *an*:: *nat* **where** $an = nat\ a$
from *a-pos* **have** $an = a$ **unfolding** *an-def* **by** *auto*
define *bn*:: *nat* **where** $bn = nat\ b$
from *b-pos* **have** $bn = b$ **unfolding** *bn-def* **by** *auto*
have $an \geq 1$ **using** ‹*int an* = *a*› ‹$a \geq 1$› **by** *auto*
moreover have $bn \geq 1$ **using** ‹*int bn* = *b*› ‹$b \geq 1$› **by** *auto*
moreover have *odd an* **using** ‹*odd a*› ‹*int an* = *a*› **by** *auto*
moreover have *odd bn* **using** *b-def* ‹*int bn* = *b*› **by** *auto*
moreover have $bn \hat{} 2 < 4 * an$ **using** *int-ineq1* ‹*int an* = *a*› ‹*int bn* = *b*› **using** *of-nat-less-iff* **by** *fastforce*
moreover have $3 * an < bn \hat{} 2 + 2 * bn + 4$ **using** *int-ineq2* ‹*int an* = *a*› ‹*int bn* = *b*› **using** *of-nat-less-iff* **by** *fastforce*
ultimately have $\exists s\ t\ u\ v :: int.\ s \geq 0 \wedge t \geq 0 \wedge u \geq 0 \wedge v \geq 0 \wedge an = s\hat{}2 + t\hat{}2 + u\hat{}2 + v\hat{}2 \wedge$
$bn = s+t+u+v$ **using** *four-nonneg-int-sum* **by** *auto*
hence $\exists s\ t\ u\ v :: int.\ s \geq 0 \wedge t \geq 0 \wedge u \geq 0 \wedge v \geq 0 \wedge a = s\hat{}2 + t\hat{}2 + u\hat{}2 + v\hat{}2 \wedge$
$b = s+t+u+v$ **using** ‹*int an* = *a*› ‹*int bn* = *b*› **by** *auto*
then obtain *s t u v* :: *int* **where** *stuv*: $s \geq 0 \wedge t \geq 0 \wedge u \geq 0 \wedge v \geq 0 \wedge a = s\hat{}2 + t\hat{}2 + u\hat{}2 + v\hat{}2 \wedge$
$b = s+t+u+v$ **by** *auto*
hence $N = (m*(s\hat{}2+t\hat{}2+u\hat{}2+v\hat{}2-s-t-u-v)\ div\ 2) + s+t+u+v + 1$ **using** *N-expr2* **by** (*smt* (*verit*, *ccfv-threshold*))
hence $N = (m*(s\hat{}2-s+t\hat{}2-t+u\hat{}2-u+v\hat{}2-v)\ div\ 2) + s+t+u+v + 1$

by (*smt* (*verit*, *ccfv-SIG*))
hence $N = (m * (s * (s-1) + t * (t-1) + u * (u-1) + v * (v-1))\ div\ 2) + s+t+u+v + 1$ **by** (*simp add*: *power2-eq-square algebra-simps*)
hence *previous-step*: $N = (m * s * (s-1) + m * t * (t-1) + m * u * (u-1) + m * v * (v-1))\ div\ 2 + s+t+u+v + 1$ **by** (*simp add*: *algebra-simps*)
moreover have $2\ dvd\ m * s * (s-1)$ **by** *simp*
moreover have $2\ dvd\ m * t * (t-1)$ **by** *simp*
moreover have $2\ dvd\ m * u * (u-1)$ **by** *simp*
moreover have $2\ dvd\ m * v * (v-1)$ **by** *simp*
ultimately have $N = m * s * (s-1)\ div\ 2 + m * t * (t-1)\ div\ 2 + m * u * (u-1)\ div\ 2 + m * v *(v-1)\ div\ 2 + s+t+u+v + 1$ **by** *fastforce*
hence *N-expr3*: $N = m * s * (s-1)\ div\ 2 + s + m * t * (t-1)\ div\ 2 + t + m * u * (u-1)\ div\ 2 + u + m * v * (v-1)\ div\ 2 + v + 1$ **by** *auto*
define *sn*::*nat* **where** $sn = nat\ s$
define *tn*::*nat* **where** $tn = nat\ t$
define *un*::*nat* **where** $un = nat\ u$
define *vn*::*nat* **where** $vn = nat\ v$
have $sn = s$ **using** *stuv sn-def* **by** *auto*
hence $m * sn * (sn-1) = m * s * (s-1)$ **by** *fastforce*
hence $m * sn * (sn-1)\ div\ 2 = m * s * (s-1)\ div\ 2$ **by** *linarith*
have $tn = t$ **using** *stuv tn-def* **by** *auto*
hence $m * tn * (tn-1) = m * t * (t-1)$ **by** *fastforce*
hence $m * tn * (tn-1)\ div\ 2 = m * t * (t-1)\ div\ 2$ **by** *linarith*
have $un = u$ **using** *stuv un-def* **by** *auto*
hence $m * un * (un-1) = m * u * (u-1)$ **by** *fastforce*
hence $m * un * (un-1)\ div\ 2 = m * u * (u-1)\ div\ 2$ **by** *linarith*
have $vn = v$ **using** *stuv vn-def* **by** *auto*
hence $m * vn * (vn-1) = m * v * (v-1)$ **by** *fastforce*
hence $m * vn * (vn-1)\ div\ 2 = m * v * (v-1)\ div\ 2$ **by** *linarith*
have $N = m * sn * (sn-1)\ div\ 2 + sn + m * tn * (tn-1)\ div\ 2 + tn + m * un * (un-1)\ div\ 2 + un + m * vn * (vn-1)\ div\ 2 + vn + 1$
using *N-expr3* ‹$sn = s$› ‹$tn = t$› ‹$un = u$› ‹$vn = v$› ‹$m * sn * (sn-1)\ div\ 2 = m * s * (s-1)\ div\ 2$› ‹$m * tn * (tn-1)\ div\ 2 = m * t * (t-1)\ div\ 2$› ‹$m * un * (un-1)\ div\ 2 = m * u * (u-1)\ div\ 2$› ‹$m * vn * (vn-1)\ div\ 2 = m * v * (v-1)\ div\ 2$› **by** *linarith*
hence $N = polygonal\text{-}number\ m\ sn + polygonal\text{-}number\ m\ tn + polygonal\text{-}number\ m\ un + polygonal\text{-}number\ m\ vn + 1$
using *Polygonal-Number-Theorem-Gauss.polygonal-number-def* **by** *presburger*
also have $polygonal\text{-}number\ m\ 1 = 1$ **using** *Polygonal-Number-Theorem-Gauss.polygonal-number-def* **by** *auto*
ultimately have $N = polygonal\text{-}number\ m\ sn + polygonal\text{-}number\ m\ tn + polygonal\text{-}number\ m\ un + polygonal\text{-}number\ m\ vn + polygonal\text{-}number\ m\ 1$ **by** *auto*
thus *?thesis* **by** *blast*
qed
show *?thesis* **using** *thm-odd-n thm-even-n* **by** *blast*
qed
qed
end

References

[1] A. Danilkin and L. Chevalier. Three squares theorem. *Archive of Formal Proofs*, May 2023. https://isa-afp.org/entries/Three_Squares.html, Formal proof development.

[2] M. B. Nathanson. *Additive Number Theory: The Classical Bases*, volume 164 of *Graduate Texts in Mathematics*. Springer, New York, 1996.