```
⊢ ∀x:ℕ. (∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))})
|
BY (DivNatInduction ⌈4⌉· THEN Auto)
|\
| ⊢ ∃r:{ℕ| (((r * r) ≤ 0) ∧ 0 < (r + 1) * (r + 1))}
| |
1 BY (With ⌈0⌉ (D 0)· THEN Auto')
 \
  1. x: ℕ+
  2. ∃r:{ℕ| (((r * r) ≤ (x ÷ 4)) ∧ x ÷ 4 < (r + 1) * (r + 1))}
  ⊢ ∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))}
  |
  BY (D (-1)
  |   THEN Auto
  |   THEN (InstLemma `div_rem_sum` [⌈x⌉;⌈4⌉]· THENA Auto)
  |   THEN (InstLemma `rem_bounds_1` [⌈x⌉;⌈4⌉]· THENA Auto))
  |
  2. r: ℕ
  [3]. ((r * r) ≤ (x ÷ 4)) ∧ x ÷ 4 < (r + 1) * (r + 1)
  4. x = (((x ÷ 4) * 4) + (x rem 4))
  5. (0 ≤ (x rem 4)) ∧ x rem 4 < 4
  ⊢ ∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))}
  |
  BY ((Evaluate ⌈r2 = (2 * r)⌉· THENA Auto)
  |   THEN (Evaluate ⌈r3 = (r2 + 1)⌉· THENA Auto)
  |   THEN (Decide ⌈x < r3 * r3⌉· THENA Auto))
  |\
  | 6. r2: ℤ
  | 7. r2 = (2 * r)
  | 8. r3: ℤ
  | 9. r3 = (r2 + 1)
  | 10. x < r3 * r3
  | ⊢ ∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))}
  | |
  1 BY (With ⌈r2⌉ (D 0)· THEN Auto')·
  | |
  | 3. (r * r) ≤ (x ÷ 4)
  | 4. x ÷ 4 < (r + 1) * (r + 1)
  | 5. x = (((x ÷ 4) * 4) + (x rem 4))
  | 6. 0 ≤ (x rem 4)
  | 7. x rem 4 < 4
  | 8. r2: ℤ
  | 9. r2 = (2 * r)
  | 10. r3: ℤ
  | 11. r3 = (r2 + 1)
  | 12. x < r3 * r3
  | ⊢ (r2 * r2) ≤ x
  | |
  1 BY (ElimVar 'r3' THEN ElimVar 'r2' THEN Auto')
   \
    6. r2: ℤ
    7. r2 = (2 * r)
    8. r3: ℤ
    9. r3 = (r2 + 1)
    10. ¬x < r3 * r3
    ⊢ ∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))}
```

```
|
BY (With ⌈r3⌉ (D 0)· THEN Auto’)·
|
3. (r * r) ≤ (x ÷ 4)
4. x ÷ 4 < (r + 1) * (r + 1)
5. x = (((x ÷ 4) * 4) + (x rem 4))
6. 0 ≤ (x rem 4)
7. x rem 4 < 4
8. r2: ℤ
9. r2 = (2 * r)
10. r3: ℤ
11. r3 = (r2 + 1)
12. ¬x < r3 * r3
13. (r3 * r3) ≤ x
⊢ x < (r3 + 1) * (r3 + 1)
|
BY (ElimVar ‘r3’ THEN ElimVar ‘r2’ THEN Auto’)·
```

Extract:

```
λx.letrec sqrt(x) =
  if x = 0 then 0
  else let z := x ÷ 4 in
       let r2 := 2 * (sqrt z) in
       let r3 := r2 + 1 in
         if (x) < (r3 * r3)  then r2
         else r3 in
  sqrt(x)
```