```
⊢ ∀x:ℕ. (∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))})
|
BY ((GeneralInductionOnNat THENA Auto) THEN CaseNat 0 ‘x’)
|\
| 1. x: ℕ
| 2. ∀x1:ℕx. (∃r:{ℕ| (((r * r) ≤ x1) ∧ x1 < (r + 1) * (r + 1))})
| 3. x = 0
| ⊢ ∃r:{ℕ| (((r * r) ≤ 0) ∧ 0 < (r + 1) * (r + 1))}
| |
1 BY (With ⌈0⌉ (D 0)· THEN Auto’)
 \
  1. x: ℕ
  2. ∀x1:ℕx. (∃r:{ℕ| (((r * r) ≤ x1) ∧ x1 < (r + 1) * (r + 1))})
  3. ¬(x = 0)
  ⊢ ∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))}
  |
  BY ((InstHyp [⌈x - 1⌉] 2· THENA Auto) THEN D (-1))
  |
  4. r: ℕ
  [5]. ((r * r) ≤ (x - 1)) ∧ x - 1 < (r + 1) * (r + 1)
  ⊢ ∃r:{ℕ| (((r * r) ≤ x) ∧ x < (r + 1) * (r + 1))}
  |
  BY ((Evaluate ⌈r2 = r⌉· THENA Auto)
  |   THEN (Evaluate ⌈r3 = (r2 + 1)⌉· THENA Auto)
  |   THEN (Decide ⌈x < r3 * r3⌉· THENA Auto))
  |\
  | 6. r2: ℤ
  | 7. r2 = 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’)
  | |
  | 5. (r * r) ≤ (x - 1)
  | 6. x - 1 < (r + 1) * (r + 1)
  | 7. r2: ℤ
  | 8. r2 = r
  | 9. r3: ℤ
  | 10. r3 = (r2 + 1)
  | 11. x < r3 * r3
  | ⊢ (r2 * r2) ≤ x
  | |
  1 BY (ElimVar ‘r3’ THEN ElimVar ‘r2’ THEN Auto’)
   \
    6. r2: ℤ
    7. r2 = 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’)
    |
    5. (r * r) ≤ (x - 1)
```

```
6. x - 1 < (r + 1) * (r + 1)
7. r2: ℤ
8. r2 = r
9. r3: ℤ
10. r3 = (r2 + 1)
11. ¬x < r3 * r3
12. (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 r2 := sqrt (x - 1) in
       let r3 := r2 + 1 in
         if (x) < (r3 * r3) then r2
         else r3 in
  sqrt(x)
```