1 Definitions and Exercises

Fill in each oops with a proof of the theorem.

```isar
primrec sum_odds :: "nat ⇒ nat" where
  "sum_odds 0 = 0" |
  "sum_odds (Suc n) = (2 * (Suc n) - 1) + sum_odds n"

theorem "sum_odds n = n * n"
oops
```

```isar
primrec sum_evens :: "nat ⇒ nat" where
  "sum_evens 0 = 0" |
  "sum_evens (Suc n) = 2 * (Suc n) + sum_evens n"

theorem "sum_evens n = n * (n + 1)"
oops
```

The notion that \( b \) divides \( a \) is written \( b \mid a \) in Isabelle, and is defined as follows.

\[(b \mid a) = (\exists k. a = b \cdot k)\]

The dvd_add theorem may come in handy:

```isar
[a dvd b; a dvd c] ⇒ a dvd b + c
```

```isar
theorem fixes n::nat shows "0 < n −→ 2 dvd (n * n + n)"
oops
```
2 Solutions

theorem "sum_odds n = n * n"
proof (induct n)
  show "sum_odds 0 = 0 * 0" by simp 
next
  fix n assume IH: "sum_odds n = n * n"
  have "sum_odds (Suc n) = (2 * (Suc n) - 1) + sum_odds n" by simp
  also from IH have "... = (2 * (Suc n) - 1) + n * n" by simp
  also have "... = Suc n * Suc n" by simp
  finally show "sum_odds (Suc n) = Suc n * Suc n".
qed

theorem "sum_evens n = n * (n + 1)"
proof (induct n)
  show "sum_evens 0 = 0 * (0 + 1)" by simp 
next
  fix n::nat assume IH: "sum_evens n = n * (n + 1)"
  have "sum_evens (Suc n) = 2 * (Suc n) + sum_evens n" by simp
  also from IH have "... = 2 * (Suc n) + n * (n + 1)" by simp
  also have "... = (Suc n) * ((Suc n) + 1)" by simp
  finally show "sum_evens (Suc n) = (Suc n) * ((Suc n) + 1)".
qed

theorem fixes n::nat shows "0 < n → 2 dvd (n * n + n)"
proof (induct n)
  show "0 < (0::nat) → (2::nat) dvd (0 * 0 + 0)"
  proof
    assume 1: "0 < (0::nat)"
    from 1 have 2: "False" by simp
    from 2 show "2 dvd (0 * 0 + 0)" .
  qed
next
  fix n::nat assume IH: "0 < n → 2 dvd (n * n + n)"
  show "0 < Suc n → 2 dvd Suc n * Suc n + Suc n"
  proof
    assume "0 < Suc n"
    show "2 dvd Suc n * Suc n + Suc n"
    proof (cases n)
      case 0
      have 1: "Suc (Suc 0) = 2 * 1" by simp
      from 1 have 2: "∃ k. Suc (Suc 0) = 2 * k" ..
      from 2 0 show "2 dvd Suc n * Suc n + Suc n" unfolding dvd_def by simp
    next
      fix m assume N: "n = Suc m"
      from N have 1: "0 < n" by simp
      from 1 IH have 2: "2 dvd (n * n + n)" by simp
      have 3: "2 * (n + 1) = 2 * (n + 1)" by simp
      from 3 have 4: "∃ k. 2 * (n + 1) = 2 * k" ..
      from 4 have 5: "2 dvd 2 * (n + 1)" unfolding dvd_def .
      from 2 5 have 6: "2 dvd ((n * n + n) + 2 * (n + 1))" by (rule dvd_add)
have 7: "Suc n * Suc n + Suc n = (n * n + n) + (2 * (n + 1))" by simp from 6 show "2 dvd Suc n * Suc n + Suc n" by (simp only: 7) qed qed qed