Problem Sheet 2

Henri Sota
h.sota@jacobs-university.de
Computer Science 2022

September 27, 2019

Problem 2.1

Let n be a natural number. If n is not divisible by 3, then n is also not divisible by 15.
This is an implication $p \to q$ where $p =$ "n is not divisible by 3", $q =$ "n is not divisible by 15"

Proof. To prove that $p \to q$, we prove the contrapositive:

$$\neg q \to \neg p$$

which is: If n is divisible by 15, then n is divisible by 3.
Assume $\neg q$. If n is divisible by 15, that means we express n in the form:

$$\begin{aligned} n &= 15 * k \qquad \text{for } k \in \mathbb{N} \\ &= (3 * 5) * k \\ &= 3 * 5 * k \\ &= 3 * (5 * k) \end{aligned}$$

By rearranging the terms, 3 appears to be a factor of n, which implies that n is divisible by 3.

■

Problem 2.2

Let n be a natural number with $n \geq 1$. Prove that the following holds:

$$1^2 + 3^2 + 5^2 + \ldots + (2n-1)^2 = \sum_{k=1}^{n} (2k-1)^2 = \frac{2n(2n-1)(2n+1)}{6} \tag{1}$$

Proof. We use induction. The induction hypothesis will be equation 1
Base Case: Our base case is $P(1)$ when $n = 1$

$$\text{Left Side of Equation} : \sum_{k=1}^{1} (2k-1)^2 = (2*1-1)^2 = (2-1)^2 = 1^2 = 1$$

$$\text{Right Side of Equation} : \frac{2n(2n-1)(2n+1)}{6} = \frac{2*1(2*1-1)(2*1+1)}{6} = \frac{2*1*3}{6} = \frac{6}{6} = 1$$

Therefore our base case has been proven to be true, because both sides of the equation equal to 1 when $n = 1$.

Induction Step: Assume $P(n)$ is true, that is equation 1 holds for some nonnegative integer n bigger than 0. Then adding $P(n+1)$ to both sides of the equation implies that

$$\begin{aligned}
1^2+3^2+5^2+\ldots+(2n-1)^2+(2(n+1)-1)^2 &= \frac{2n(2n-1)(2n+1)}{6}+(2(n+1)-1)^2 \\
&= \frac{2n(2n-1)(2n+1)}{6}+(2n+1)^2 \\
&= \frac{2n(2n-1)(2n+1)+6(2n+1)^2}{6} \\
&= \frac{[2n(2n-1)+6(2n+1)](2n+1)}{6} \\
&= \frac{4n^2-2n+12n+6)(2n+1)}{6} \\
&= \frac{(4n^2+10n+6)(2n+1)}{6} \\
&= \frac{2(2n^2+5n+3)(2n+1)}{6} \\
&= \frac{2(2n+3)(n+1)(2n+1)}{6} \\
&= \frac{(2n+2)(2n+1)(2n+3)}{6} \\
&= \frac{(2(n+1))(2(n+1)-1)(2(n+1)+1)}{6}
\end{aligned}$$

By using a substitute variable a for $n+1$, our equation becomes

$$1^2+3^2+5^2+\ldots+(2n-1)^2+(2(n+1)-1)^2 = \frac{(2a)(2a-1)(2a+1)}{6}$$

which proves $P(n+1)$.
So it follows by induction that $P(n)$ is true for arbitrary nonnegative integers n such that $n \geq 1$.
■

Problem 2.3

Algorithmic steps to determine whether a year is a leap year:

1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
4. Year is a leap year.
5. Year is not a leap year.

First, we need to check if year is divisible by 4. If that is False, we conclude that year is not a leap year. (ex. 3) If that is True, we need the special cases: when year is divisible by 100 and when year is divisible by 400. If the year is not divisible by 100, we conclude that it is a leap year. (ex. 72) If the year is divisible by 100, we need to check if it is also divisible by 400. If last one is False, we conclude that year is not a leap year. (ex. 1900) If last one is True, year is a leap year. (ex. 2000)

a)
```
isLeapYear :: Int -> Bool
isLeapYear year = (mod year 4 == 0) && (not (mod year 100 == 0) || (mod year
    400 == 0))
```

b)
```
isLeapYear' :: Integer -> Bool
isLeapYear' year
    | isDivisibleBy 400 = True
    | isDivisibleBy 100 = False
    | isDivisibleBy 4 = True
    | otherwise = False
        where isDivisibleBy num = mod year num == 0
```

In order to test these two functions, I've used multiple calls to these functions with different input, namely 0, 1, 4, 72, 1900, 2000, which respectively produced True, False, True, True, False, True.

Problem 2.4

a)
```
-- Pattern matching with a recursive function that calls itself with arguments
    : an Int that decrements each time going through recursion and a list of
    Chars (String)  that takes the first element of the list and puts it at
    the end of the list in order to rotate on the left
rotate :: Int -> [a] -> [a]
rotate 0 string = string
rotate n string = rotate (n-1) ((drop 1 string) ++ (take 1 string))
```

b)
```
-- List comprehension that applies function rotate each time with argument i
    that goes from 0 to length of string - 1
circle :: [a] -> [[a]]
circle string = [((rotate i string)) | i <- [0..(length string - 1)]]
```