

ASSIGNMENT 5B – SEQUENCE COMPREHENSIONS

Advanced programming paradigms

In this assignment, you will apply your new knowledge on tuples and use sequence comprehensions (*for*) to produce an exhaustive search for postcards and to work with the n queens problem.

Question 1 – Tuples

This is the code given in class for returning the sums of the couple of integers under a certain value that are prime:

```
1 def primeSum(max: Int): List[(Int, Int)] =
2   for {
3     i <- (1 to max).toList
4     j <- (1 to max).toList
5     if (isPrime(i + j))
6   } yield (i, j)
7
8 def isPrime(i: Int): Boolean =
9   i match {
10    case i if i <= 1 => false
11    case 2 => false
12    case _ => !(2 to (i - 1)).exists(x => i % x == 0)
13  }
```

The problem of this code is that the `primeSum` function returns values twice. For instance, it returns $(1, 2)$ but also $(2, 1)$. Because those values are only permuted, we would like to remove redundant pairs. Without changing the existing code, devise a function that removes the existing permutations from a similar list.

- (a) Using pattern matching with tuples and recursion
- (b) Using folding with tuples

Question 2 – Sending postcards

In this exercise, you will use `for` comprehensions for generating postcards when you are travelling. You are given the following code that codes defines the city you visit, the person you want to send the cards to and the travellers that are sending postcards.

```
1 val cities = List("Paris", "London", "Berlin", "Lausanne")
2 val relatives = List("Grandma", "Grandpa", "Aunt Lottie", "Dad")
3 val travellers = List("Pierre-Andre", "Rachel")
```

- (a) In each city you visit, each travellers send a postcard to all their relatives. Because the task is tedious, you are asked to write a program that generates the text for the postcards as follows:

```
Dear Grandma, Wish you were here in Paris! Love, Pierre-Andre
Dear Grandma, Wish you were here in London! Love, Pierre-Andre
...
Dear Grandpa, Wish you were here in Paris! Love, Pierre-Andre
Dear Grandpa, Wish you were here in London! Love, Pierre-Andre
...
Dear Grandma, Wish you were here in Paris! Love, Rachel
Dear Grandma, Wish you were here in London! Love, Rachel
...
Dear Dad, Wish you were here in Lausanne! Love, Rachel
```

- (b) How do you modify your code in order to send postcards only to the family members whose name start with the letter G? Your answer can (and should) be written in one line.

Question 3 – *Printing the n queens solutions*

A possible solution to the n -queen problem that your had to analyze is as follows:

```

1  def queens(n: Int): List[List[(Int, Int)]] = {
2    def placeQueens(k: Int): List[List[(Int, Int)]] =
3      if (k == 0)
4        List(List())
5      else
6        for {
7          queens <- placeQueens(k - 1)
8          column <- 1 to n
9          queen = (k, column)
10         if isSafe(queen, queens)
11         } yield queen :: queens
12
13    placeQueens(n)
14  }
15
16  def isSafe(queen: (Int, Int), queens: List[(Int, Int)]) =
17    queens forall (q => !inCheck(queen, q))
18
19  def inCheck(q1: (Int, Int), q2: (Int, Int)) =
20    q1._1 == q2._1 || // same row
21    q1._2 == q2._2 || // same column
22    (q1._1 - q2._1).abs == (q1._2 - q2._2).abs // on diagonal

```

Using for comprehensions (but not only that), write a function `def printChessBoard(...)` that returns a string containing all the solutions for a given n . For instance,

```
1  println(printChessBoard(queens(4)))
```

should display the following on the console:

```

Solution 1:
|_|♚|_|_|
|_|_|_|♚|
|♚|_|_|_|
|_|_|♚|_|

Solution 2:
|_|_|♚|_|
|♚|_|_|_|
|_|_|_|♚|
|_|♚|_|_|

```

Remark: Note that the Unicode character for the black queen used in the above example is `\u265b`.