CPS2001
Programming Paradigms Assignment 2
Functional Paradigm

Mark Galea
mgale11@um.edu.mt

31st October 2016

The aim of this assignment is to strengthen your understanding of the functional programming paradigm. You are given four separate tasks to be implemented using the Scala programming language.

1 Preliminaries

This is the second assignment for CPS2001; two more will follow with time. It carries 10% of the final mark and hence will be scored out of 10 points as follows:

- **1 point** for Task 1
- **3 points** for Task 2, as follows
  - **1 point** for Part A,
  - **2 points** for Part B,
- **2 points** for Task 3
- **4 points** for Task 4

Parts of the assignment may also be present in the exam. The deadline for submission of the complete assignment is at **NOON** on **21st of November 2016** and it must be carried out **individually**.

You are reminded that we take plagiarism seriously. While you are encouraged to discuss ideas with fellow students and perform additional research you cannot copy and steal ideas. Read and follow the information

found at `https://www.um.edu.mt/ict/Plagiarism` carefully. Extreme measures might be taken in case of plagiarism. Note that you might be randomly selected for a 10 minute interview to discuss your solution.

2 Notes

Create a Scala Project in your preferred IDE such as Eclipse or IntelliJ. Create a package `recfun` and copy the `Main.scala` file from the assignment resources on VLE.

1. **Do not** change the given function definitions.
2. **Do not** add additional files except for your own test cases. Testing is highly recommended!
3. **Do not** use any software packages or libraries that provide out-of-the-box solutions.
4. **Do not** use the `var` keyword since this implies mutation of state.
5. **Do not** use loops or other iterative constructs. Use recursive functions.
6. Comment your code when necessary but **do not** comment the obvious.

Note that most of these functions can be implemented in less than 15 lines of code, if this is not the case consider revising your solution.

3 Tasks

3.1 Task 1 - Pascal's Triangle

Write a function that computes the elements of Pascal's triangle. Do this exercise by implementing the following function in `Main.scala`

```
def pascal(c: Int, r: Int): Int = ???
```

This function takes a column `c` and a row `r`, counting from **0** and returns the number at that spot in the triangle. For example, `pascal(0,0)=1`, `pascal(1,4)=4` and `pascal(2,3)=3`.

3.2 Task 2 - Fibonacci Numbers

Write a simple function that computes the fibonacci sequence. Do this exercise by implementing the relevant function in `Main.scala`

3.2.1 Part A

```
def fib(n: Long): Long = ???
```

This function takes an index `n` and returns the number in the fibonacci sequence at that index. Index is zero based. For example, `fib(0)=0`, `fib(1)=1`, `fib(2)=1`, `fib(10)=55`.

3.2.2 Part B

The previous implementation is correct but quite inefficient - try calculating `fib(50)` for example and see what happens. The inefficiently stems from the fact that to calculate `fib(n)` we are required to calculate both `fib(n-2)` and `fib(n-1)` and to calculate in turn `fib(n-1)` we would have to calculate `fib(n-2)` again.

Using a tuple we can give an efficient solution to the problem. The next value in the sequence is given by adding the previous two, so what we do is to write a function which returns two consecutive values as the result. This is the idea behind the fast fibonacci implementation. Do this exercise by implementing the following function in `Main.scala`

```
def fastFib(x: Long ): Long = ???
```

Hint: Consider implementing first the inner function below which, given a tuple `x` *with components* `(x._1, x._2)` *we can obtain the next pair as* `(x._2, x._2 + x._1)`*, which is the effect of the fibStep function.*

```
def fibStep(x:(Long, Long)): (Long, Long) = ???
```

To make sure you got the correct implementation consider trying out the following test cases

Test	Value
fastFib(30)	832040L
fastFib(40)	102334155L
fastFib(50)	12586269025L
fastFib(83)	99194853094755497L

3.3 Task 3 - Parenthesis Balancing

Write a function which verifies the balancing of parentheses in a string, which we represent as a `List[Char]` not a `String`. Do this exercise by implementing the following function in `Main.scala`

```
def balance(chars: List[Char]): Boolean = ???
```

To make sure you got the correct implementation consider trying out the following test cases

Test	Value
balance("(if (zero) (x + 1) else (x * 3))".toList)	true
balance("(a * (b + b)* c)".toList)	true
balance"(if (weekend) :) else :(".toList)	false
balance("(()))(".toList)	false

3.4 Task 4 - Counting Change

Write a function that counts how many different ways you can make change for an amount, given a list of coin denominations. For example, there are 3 ways to give change for 4 if you have coins with denomination 1 and 2: 1+1+1+1, 1+1+2, 2+2.

Do this exercise by implementing the following function in `Main.scala`

```
def countChange(money: Int, coins: List[Int]): Int = ???
```

To make sure you got the correct implementation consider trying out the following test cases

Test	Value
countChange(4,List(1,2))	3
countChange(300,List(5,10,20,50,100,200,500))	1022
countChange(301,List(5,10,20,50,100,200,500))	0
countChange(300,List(500,5,50,100,20,200,10))	1022

4 Report

This assignment *may* to be accompanied by a concise report outlining the remarks, decisions or assumptions you have made while carrying out the tasks. Note that this is **not assessed** but treat it as a means for you to communicate your thoughts in writing.

5 Submissions

All submissions have to take place by the stipulated deadline.

1. Submit a hard copy of the report *(if any)* and the plagiarism declaration form to Ms. Vanessa Borg.

2. Package all the source code and submit the assignment through VLE.

3. Make sure that the files and functions are named exactly as specified in this document.

6 Conclusion

If you have reasonable questions please contact me on my email address. Obviously I will not provide implementation details however if you require some clarification feel free to contact me. Dedicate enough time for completing this assignment since no **extensions** will be provided; start early and ideas will mature. Good luck!