Department of Computer Science

COS284 Practical and Assignment 4: Functions and Arrays

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

This document contains both Practical 4 and Assignment 4. In general the Assignments will build upon the work of the current Practical.

1.1 Submission

The Practical will be fitchfork only and is due at 20:00 on Tuesday the 10th of September. The Assignment will also be fitchfork only and is due at 17:00 on Friday 20th of September. You may use the practical sessions to work on your assignment in the labs and ask for assistance if it is required.

1.2 Plagiarism policy

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's work or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result.

For more on the University of Pretoria's plagiarism policy, you may visit the following webpage: `http://www.library.up.ac.za/plagiarism/index.htm`

1.3 Practical component [25%]

You must first complete both tasks of this practical. Once you have done so you are required to submit files to fitchfork for marking.

1.3.1 Task 1: Spiral Reading[10 %]

For this task you must implement an assembler function that performs a spiral read of a matrix ($N \times M$ matrix with $N \geqslant 1$ and $M \geqslant 1$) with the following format `void spiralRead(char**, int rows, int cols)`

Example of the output should be as follows: Given a matrix as such:

$$\begin{bmatrix} '1' & '2' & '3' \\ '8' & '9' & '4' \\ '7' & '6' & '5' \end{bmatrix}$$

You should produce the following output:

```
'123456789'
```

You will be given an **main.c** that contains a function called `printChar(char)` to print a character for you (use this within your `spiralRead(char**,int,int)` function) and a sample **makefile**
When you are finished, create a tarball containing your source code file named **spiralRead.asm** and upload it to the assignments.cs.up.ac.za website, under the **Practical 4 Task 1** submission slot.

1.3.2 Task 2: Recursion [15%]

For this task you must implement a recursive assembler function `int64_t funct(int64_t t)` to do the following:

- if `t==0` then `return 1`
- if `t==1` then `return 2`
- otherwise `return funct(t-1) - funct(t-2)*(t-1)`

Example of the output should be as follows:

- Given 6
- `funct` should produce 36

The function should return a 64-bit integer value.
You will be given a **main.c** that will call your `funct(int64_t)` function. You may assume that $t \geqslant 0$. When you are finished create a tar ball containing your source code file named **funct.asm** and upload it to the assignments.cs.up.ac.za website, under the **Practical 4 Task 2** submission slot.

1.4 Assignment component [75%]

For this assignment you will be required to implement several functions that will form a complete implementation of the Vigenere cipher. To aid you in this here is a description of the operation of the cipher itself `https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher`. Remember that in this particular cipher white space is consumed, thus HELLO WORLD must be converted to HELLOWORLD before you encrypt it. In addition you may assume for all tasks that input will always be uppercase letters and there will be no punctuation.

1.4.1 Task 1: Generate a matrix [15%]

For this task you must implement an assembly function called `char** populateMatrix()` that will allocate and populate a 26x26 matrix of characters. Each row of this matrix will consist of the alphabet shifted left based on the row index (i.e. row 0 is unaltered, row 1 is shifted left by 1, row 2 is shifted left by 2 etc.). The function must return a pointer to the matrix.

When you are finished, create a tarball containing your source code file named **populateMatrix.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 4 Task 1** submission slot.

1.4.2 Task 2: Character Encryption [25%]

For this task you must implement an assembly function called `char encryptChar(char**, char, char)` which takes a `char**` (the matrix from task 1) and two `char`s as input and encrypts the first `char` using the matrix and the key value provided as the second `char` as described in the specification of the cipher. The function must return the encrypted `char`.

Example:

```
Given the input char: A
Given the key char: L
Should produce the ciphertext: L
```

When you are finished, create a tarball containing your source code file named **encryptChar.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 4 Task 2** submission slot.

1.4.3 Task 3: String encryption [35%]

For this task, you must implement an assembly function called `char* encryptString(char**, char*, char*)` that takes the matrix returned from `populateMatrix()` and two string values (plaintext and keyword) and produces a keyword array and using the corresponding letters from the plaintext and keyword array, repeatedly calls the function defined in task 2 to encrypt the string. The function must then return the fully encrypted text.

Example:

```
Given input string: ATTACK AT DAWN
Given the keyword: LEMON
Should produce the ciphertext: LXFOPVEFRNHR
```

When you are finished, create a tarball containing your source code file named **encryptString.asm** and upload it to the assignments.cs.up.ac.za website, under the **Assignment 4 Task 3** submission slot.

1.5 A note on the assignment tasks:

For all assignment tasks you will be given a C wrapper class that will handle input and output for you. You must simply implement the functions that will be called from inside the class and provide the correct return values.

2 Mark Distribution

Activity	**Mark**
Prac Task 1	10
Prac Task 2	15
Assignment Task 1	15
Assignment Task 2	25
Assignment Task 3	35
Total	**100**