Department of Computer Science

COS284 Practical and Assignment 3: Branching and Looping

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

This document contains both practical 3 and assignment 3. In general the assignments will build upon the work of the current practical.

1.1 Submission

The practicalneeds to be submitted on Fitchfork under Practical 3 by **24 September 22:00**. The Assignment will also need to be submitted on Fitchfork, but it will be under Assignment 3 and due by **1 October 22:00**. You may use the practical sessions to 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 all 3 tasks of this practical. Once you have done so you are required to submit and archive with all 3 tasks **task1.asm, task2.asm task3.asm** to fitchfork for marking.

1.3.1 Task 1: String length[5 %]

For this task you must implement an assembler program that does the following:

- Display "Please enter a string: "
- Store the string provided as input and determine its length.
- Display "The length of the string is: x' Where x is the length of the string. You have to account for multiple digit outputs.

Example of the output should be as follows:

```
Please enter a string: The quick brown fox was arrested for grand larceny
The length of the string is: 50
```

1.3.2 Task 2: Case swap [8%]

For this task you must implement an assembler program that does the following:

- Display "Please input a string: "
- Take the value and store it. For every upper case letter convert it to a lower case letter and vice versa.
- Display "The new string is: xxxxxxxxxxxxxxxxxxx " where xxxxxxxxxxxxxxxxx is the newly converted string.

Example of the output should be as follows:

```
Please input a string: Hello World
The new string is: hELLO wORLD
```

Note that you need only account for whitespace, no punctuation will be tested.

1.3.3 Task 3: Sum of individual numbers [12 %]

For this task you must implement an assembler program that does the following:

- Display "Please input an integer: "
- Get the input from the user

- Calculate the sum of the integer as follows: say 5 is entered, then the sum of the integer is 5+4+3+2+1.
- Display "The total sum is: y" where y is the result. (The result will not necessarily be any specific number of characters, you should accomadate variable sized inputs and outputs).

Example of the output should be as follows:

```
Please input an integer: 7
The total sum is: 28

Please input an integer: 12345
The total sum is: 76205685
```

You do not have to worry about the user entering anything aside from numbers. We will have only non-idiot users test your code, this goes for the other tasks in this practical as well.

1.4 Assignment component [75%]

For this assignment you will be required to implement two variations of the Caesar cipher, an encryption technique involving shifting letters within the alphabet. You must complete all 3 tasks in teh Assignment. Once you have done so you must create an archive with all 3 tasks **task1.asm, task2.asm, task3.asm** and submit it to ff.cs.up.ac.za under Assignment 3.

1.4.1 Task 1: Basic Caesar Cipher [20%]

For this task you must implement a basic Caeser cipher which will only take lower case letters and whitespace as input. It will also have to take a variable n as input where n is the degree to which each letter should be shifted. E.g. If n is 5 then each letter should be shifted 5 positions to the right in the alphabet, thus if n is 5 'a' would become 'f'.

Example:

```
Please input the shift degree: 05
Please input the string to encode: hello world
Encoded text: mjqqt btwqi
```

1.4.2 Task 2: Basic Decoder [10%]

For this task you must implement a decoder for the cipher you programmed in task 1. As input it will take the same shift degree as was given for task 1 and the cipher text produced by task one. It will then have to output the plain text (the phrase that was given as input to task 1)

Example:

```
Please input the shift degree: 05
Please input the string to decode: mjqqt btwqi
Decoded text: hello world
```

1.4.3 Task 3: Complex Caesar Cipher [45%]

For this task, similarly to the first task, you must take a degree of shift and a string to encode as input. There are however a few alterations to this version of the program. First your input will no longer be restricted to lower case letters and whitespace only, now you will have to account for punctuation (commas and sentence-ending punctuation only) and upper case letters as well. Secondly and more importantly you will now have to alternate shift directions. In other words if the shift factor is 5 and you have two a's in succession they will change from 'aa' to 'fv' as the first a is shifted 5 positions right and the second is shifted 5 positions left(and wraps around to the end of the alphabet).

Example:

```
Please input the shift degree: 05
Please input the string to encode: Hello World!
Encoded text: Mzqgt Rtmqy!
```

1.5 A note on the assignment tasks:

For all assignment tasks you may assume that you will not be given any numbers as part of the input to be encoded and that the shift degree will only be an integer between 0 and 26 non-inclusive.

2 Mark Distribution

Activity	**Mark**
Prac Task 1	5
Prac Task 2	8
Prac Task 3	12
Assignment Task 1	20
Assignment Task 2	10
Assignment Task 3	45
Total	**100**