Department of Computer Science

COS284 Practical and Assignment 4: Functions

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 17:00 on Friday 2 October, whereas the assignment will also be fitchfork only and due at 17:00 on Friday 9 October. You may use the Discord server to ask for assistance with the assignment and practical components.

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 [28%]

1.3.1 Task 1: Count Words [14%]

For the first task of the practical you will be required to implement a function *int64_t countWords(char*, int length)* which will count the number of words in a string. All words in the string will be separated by a space.
For example:

char c[44] = "the quick brown fox jumped over the lazy dog";
int num = countWords(c,44);

Would return: 9

You will be given a **main.c** file to test your implementation and a sample makefile.

1.3.2 Task 2: Rotate a string [14%]

For this task you will be required to implement a function *char* rotateString(char*, int length, int degree)* That will shift the characters of a string left by a certain degree and return the shifted string. For example:

char c[44] = "the quick brown fox jumped over the lazy dog";
char* num = rotateString(c,44,5);

Should return: "uick brown fox jumped over the lazy dogthe q"
You will be given a **main.c** file to test your implementation and a sample makefile.

1.4 Assignment component [72%]

For this assignment you will be implementing an in-place sorting algorithm that sorts a string of words (such as "CAT BAT HAT") into alphabetical order.

You may work under the following assumptions:

1. The size of the words in a single string will be constant.
   (a) For example, in a string "REAL FEEL DEAL..." the words will always be 4 letters and the word length will be given as a parameter to each function.
2. The number of words in the string will be given.
3. Words within the string will always be separated by a space.
4. Words will be either in all CAPS or in all lowercase.
5. You may not use C functions within your Assembly Code unless otherwise stated.

The choice of Sorting Algorithm is up to you and the only requirement is that the algorithm be one that uses in-place sorting.

Note that while words that begin with the same letter may be tested, you **do not** hove to cater for them by comparing the following letters in the respective words, rather let them apear in the order that they were given.

1.4.1 Task 1: Swap Letters [5%]

For this task you are required to implement the following function:

char swapLetters(char* string, int indexA, int indexB)*

The function takes in a string of letters and two integers that point to letters that need to be swapped and returns the string with the swap being executed on the string.

For example, calling the function in C like this:

```
char string[] = "A X Z Y D C";
printf("Before swapLetters: \t%s\n", string);
char * result = swapLetters(string, 2, 10);
printf("After swapLetters : \t%s\n", result);
```

Should Result in the following output:

```
Before swapLetters: A X Z Y D C
After swapLetters: A C Z Y D X
```

[Hint: It would be beneficial to use the *movsb* instruction]

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

1.4.2 Task 2: Swap Words [15%]

For this task you are required to implement the following function:

char swapWords(char string[], int element1, int element2, int wordSize);)*

The function will take in a string of words, two element values (integers) and the size of the words and should return the string with a swap having been completed. (The swapping should be done on whole words, of which the size of each word is constant)

To calculate the 'actual index' of the word, i.e. the index of where the word starts in the string, from the 'element number' is simply the element number multiplied by the length of a single word plus 1 (to account for the space).

That is:

$$elementNumber * (lengthOfWord + 1)$$

You have been provided with a C function called *calculateIndexOfWord* in the **main.c** that does this calculation for you. You may use this in the tasks that follow as an external function in your ASM.

As an example, calling the function in C like this:

char words[] = "CAT FAT BAT HAT RAT";
printf("Before swapWords: $\backslash t\%s\backslash n", words);$
$char*\ result\ =\ swapWords(words, 4, 1, 3);$
$printf("After\ swapWords:\ \backslash t\%s\backslash n",\ result);$

Should give you the following output:

Before swapWords: CAT FAT BAT HAT RAT
After swapWords: CAT RAT BAT HAT FAT

Note: The length of the words can be much larger then 3 and the example above is just for demonstration.

[Hint: It would be beneficial to use the *movsb* instruction coupled with the *rep* instruction]

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

1.4.3 Task 3: Sorting Algorithm [52%]

For this task you are required to implement the following function:

char sort(char string[], int numberOfWords, int lengthOfWord);*

The function takes in a string consisting of words, the total number of words and the length of a word and should return the string sorted in alphabetical order.

As an example, calling the function in C like this:
char words[] = "CAT FAT BAT HAT RAT";
printf("Before sort: $\backslash t\%s\backslash n", words);$
$char*\ result\ =\ sort(words, 5, 3);$
$printf("After\ sort:\ \backslash t\%s\backslash n",\ result);$

Should give you the following output:

Before sort: CAT FAT BAT HAT RAT
After sort: BAT CAT FAT HAT RAT

You may use the *swapWords* function from Task 2 however that is not compulsory. As stated above, the choice of sorting algorithm is entirely up to you.

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

2 Mark Distribution

Activity	**Mark**
Practical	28
Assignment	72
Total	**100**