UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS132 - Imperative Programming
Practical 7

1 Introduction

Deadline: 1st of June, 20:00

1.1 Objectives and Outcomes

The objective of this practical is to demonstrate far more complex behaviours in programming. Specifically, this is to learn about the potential of arrays and looping. Additionally, this will also cover some basic algorithms to do both sorting and searching of arrays, both tasks that are pivotal to any programming language.

This practical will consist of 3 Activities and you will be required to complete all of them as part of this practical. You are also advised to consult the Practical 1 specification for information on aspects of extracting and creating archives as well as compilation if you need it. Also consult the provided material if you require additional clarity on any of the topics covered in this practical.

Finally note that the use of C++11 functions and methods will not be allowed except for cases where its usage is explicitly mentioned.

1.2 Structure of the Practical

Each Activity is self contained and all of the code you will require to complete it will be provided on the appropriate Activity slot.

Each Activity will require you submit a separate archive to an individual upload slot. That is, each separate Activity will require its own answer archive upload. You will upload Activity 1 to Practical3_Activity1 and so on.

1.3 Submission

Submit your code to Fitchfork before the closing time. Students are **strongly advised** to submit well before the deadline as **no late submissions will be accepted**.

Also note that file names are case sensitive. Failure to name the files as specified will result in no marks being awarded.

1.4 Plagiarism

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone elses work without consent, copying a friends work (even with consent) and copying textual material from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to **http://www.ais.up.ac.za/plagiarism/index.htm** (from the main page of the University of Pretoria site, follow the *Library* quick link, and then click the *Plagiarism* link). If you have questions regarding this, please ask one of the lecturers, to avoid any misunderstanding.

1.5 Mark Distribution

Activity	**Mark**
Advanced Arrays	10
Linear Search	15
Bubble Sort	15
Total	**40**

2 Practical Activities

2.1 Activity 1: Advanced Arrays

For this activity, you are required to provide files called **act1.cpp** as well as a makefile to compile and run it. In your file, act1.cpp, should have a skeleton of a main program as per normal which you will then fill it in as per the following.

The objective of this activity is to demonstrate advanced array use, in terms of how to use declare and use them for programs.

At the start of your program, you are going to need to declare a 4 by 4 element two dimensional array. That is, 4 dimensions by 4 dimensions for a total of 16 elements in the array. Your program needs to read in from a file, **values.txt**, where each line of the file will have a comma delimited list of integer values with 4 lines in total.

Each value of each line should be placed into the array, with the line corresponding to the row index. The first value in the line goes into the first, index 0, column of the array. So the array is filled row by row, from left to right.

Once furnished, you need to compute the following:

- Row Totals: The sum of all elements per row
- Column Totals: The sum of all elements per column
- Array Average: The average of all elements in the array

The output format should be displayed as follows (with example values):

```
Row Total 1: 4
Row Total 2: 4
Row Total 3: 4
Row Total 4: 4
Col Total 1: 4
Col Total 2: 4
Col Total 3: 4
Col Total 4: 4
Array Average: 1
```

You will have a maximum of 10 uploads for this task. Be sure to include a blank **values.txt** in your submission. The string, fstream and sstream libraries are recommended.

2.2 Activity 2: Linear Search

For this activity, you are required to provide files called **act2.cpp** as well as a makefile to compile and run it. In your file, act2.cpp, should have a skeleton of a main program as per normal which you will then fill it in as per the following.

The objective of this activity is to demonstrate the linear search algorithm for arrays.

You are going to implement this as a function with the following definition:

```
string linSearch(double arr[10], double search[8]);
```

This function will represent the core functionality of your program. This function will take two arrays as arguments, both of specified size. The first array will be the array that should be searched. It will contain double values. The second array will represent a list of values to search for in the first array.

Your function will return a string, comma-delimited, which represents the indices of the searched for elements in the array. The first index in the string should correspond to the first element in the search array and so on. If the array is not found, it should show NA. For example:

```
Consider the case of arr={1.1,1.2.1.3,1.4,1.5,1.6,1.7,1.8.1.9,1.11}
and search={1.3,2.1,1.11}

Then the return value of linSearch should be: 2,NA,9
```

The search array will be of size 8, the above example is just to demonstrate what sort of results should be expected. Once implemented, the rest of your main program should be as follows. Read in from a file called **search.txt**. In this file, the first line will be a comma delimited list of double values, corresponding to arr. The second line will contain a comma delimited list of values to search for. The output should display each value (of the search results) on a new line.

You will have a maximum of 10 uploads for this task. The string, sstream and fstream library will be useful here. Be sure to include a blank **search.txt** in your submission.

2.3 Activity 3: Bubble Sort

For this activity, you are required to provide files called **act3.cpp** as well as a makefile to compile and run it. In your file, act3.cpp, should have a skeleton of a main program as per normal which you will then fill it in as per the following.

The objective of this activity is to demonstrate the bubble sort algorithm for arrays.

You are going to implement this as a function with the following definition:

```
string bubSort(double arr[8]);
```

Your program will need to read in from a file called **list.txt**. On each line, will be a comma-delimited list of integer values. Each list will be 8 elements long. Your objective is to sort each of these lists using the bubble sort algorithm.

Each line should be sorted, and then displayed in sorted (ascending order with the smallest element at the start of the string) one after the other. Each will be on a new line. As a hint, implementing a function to do the bubble sort itself will make the overall program easier to implement.

For example:

```
Given a list: 6.4,3.25,7.5,2.5,1.1,11.6,0.5

The outcome of the sorting should be: 0.5,1.1,2.5,3.25,6.4,7.5,11.6
```

You will have a maximum of 10 uploads for this task. The string, fstream and sstream libraries are recommended. Be sure to include a blank **list.txt** in your submission.