



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS132 - Imperative Programming

Practical 8

Copyright © 2021 by Emilio Singh. All rights reserved.

1 Introduction

Deadline: 25th of June, 20:30

1.1 Objectives and Outcomes

The objective of this practical is to expand on the work done in all prior practicals and introduce more complicated scenarios involving pointers and dynamic memory.

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

This practical will consist of 2 Tasks 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.

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 else's work without consent, copying a friend's 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
Arrays and Pointers	15
2D Matrices	15
Total	30

2 Practical Tasks

2.1 Task 1: Arrays and Pointers

For this task, you are required to provide files called `arrP.cpp` as well as a makefile to compile and run it. In your file, `arrP.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 task is to demonstrate the use of pointers for creating dynamic arrays. Specifically, through the use of pointers, it is possible to create arrays of a dynamic size. That is, to create an array of a specific size that can be altered as needed. The type should be integer.

Your program will need to read from a file called **values.txt**. This file has the following format:

```
4;1,2,3,4
3;6,7,9
```

The above refers to a line with two components: size and the values of an array. The first number before the semi-colon, refers to the size of the array. What follows afterwards are the values that should go in that array. The number of lines you will have to read is not specified.

You will read from this file, line by line, each of the lines, and then store the values in the array. Then the following operation should be done:

- If the largest element in the array is odd, square every value in the array and display the array, as a single comma delimited line with a new line at the end.
- If the largest element in the array is even, multiply every value by the largest value in the array and display the array, as a single comma delimited line with a new line at the end.

An example of the output given the above file would be:

```
4,8,12,16
36,49,81
```

2.2 Task 2: 2D Matrices

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

A matrix is a 2-dimensional data structure, similar to an array, but composed of both rows and columns. The structure has many uses, especially in mathematical operations. This matrix should be declared to hold integer values. It should be dynamically allocated and the size specification will only be provided inside the file so be sure to allocate your pointers accordingly.

For this exercise, you need to read from a file called **values.txt**. This file has the following format:

```
3,3
1,1,1
1,1,1
1,1,1
```

The first line of this file indicates how many rows and columns the matrix structure has which are the first and second values respectively. Each line of the textfile represents a row, and each value (separated by a comma) is a value at the specific column.

Once the values are stored in the array, you will need to determine the following:

- Determine the number of odd numbers
- Determine the number of even numbers
- Find the largest number in the matrix
- Find the smallest number in the matrix

You will display the results as follows:

```
Count Odd: 9
Count Even: 0
Largest Number: 1
Smallest Number: 1
```

3 Submission Requirements

Your submission requirements for this task are:

- makefile
- `arrP.cpp`

- mat.cpp
- values.txt

The following libraries are recommended:

- fstream
- sstream
- string

By default the use of c++11 functions is not enabled. Please take this into account when designing your programs.

Please note that marks will be subtracted from programs that have too many lines output. You are advised to make use of `while (getline (..))` to read your files and use an if statement to check if the files actually exist before attempting to open them.

You will have a maximum of 15 uploads for this practical. One submission will evaluate all tasks.