UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS110: Practical 2

Deadline: Friday 19 August 2022, 11:59

Objectives

The aim of this practical is to learn how to implement classes with dynamic arrays.

Instructions

Complete the task below. Certain classes have been provided for you in the *files* zip archive of the practical. You have also been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Upload **only** the given source files with your changes in a zip archive before the deadline. Please comment your name **and** student number in at the top of each file.

Task 1: [25]

Hints:

- Remember that the final item in the array will not be a standard integer, but will instead be an object of an integer, thus the **new** keyword will come in very handy.
- Remember that in order to access the data of a pointer object, one must first dereference it with the * operator.
- Remember that when copying over data from one pointer to another, one should take extra care to copy over the dereferenced content and not the pointer address instead. Keep this in mind when resizing.

Implement the following methods according to the given specification:

`Matrix(int in_rows, int in_columns)`

A constructor that takes in the amount of rows and columns for the matrix. If either the amount of rows or columns are less than 1, then you should initialize the matrix to a *3 x 3* matrix of 1's. If the rows and columns are more than 1, then create a matrix of the specified size and initialize all elements to 0's.

`~Matrix()`

The destructor for the matrix, correctly deallocate all memory involved with the matrix, taking care to delete object by object and not just the entire matrix.

`void resizeRows(int num_rows)`

Resize the rows of the matrix. If the new number of rows is smaller, then remove the elements that were previously there. If the new number of rows is larger, then fill the newly created rows with 0's. If the number of rows given is less than 1, output the error message "**Error: cannot have less than 1 row**" followed by an end line character.

`void resizeColumns(int num_cols)`

Resize the columns of the matrix. If the new number of columns is smaller, then remove the elements that were previously there. If the new number of columns is larger, then fill the newly created columns with 0's. If the number of columns given is less than 1, output the error message "**Error: cannot have less than 1 column**" followed by an end line character.

`void setMatrix()`

A function to fill the given matrix. You should make use of **cin** in order to obtain the data. The order of the elements taken in should fill the matrix in row order. Thus if the input for a *3 x 3* matrix is 1,2,3,4,5,6,7,8,9. It will create the matrix.

1 2 3

4 5 6

7 8 9

Submission

You need to submit your source files on the Assignment website `https://ff.cs.up.ac.za/`. All tasks need to be implemented (or at least stubbed) before submission. Place **matrix.cpp** file in a zip or tar/gzip archive (you need to compress your tar archive) named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. You have 5 days to complete this practical, regardless of which practical session you attend. Upload your archive to the *Practical 1* slot on the Assignment website. Submit your work before the deadline. No late submissions will be accepted.