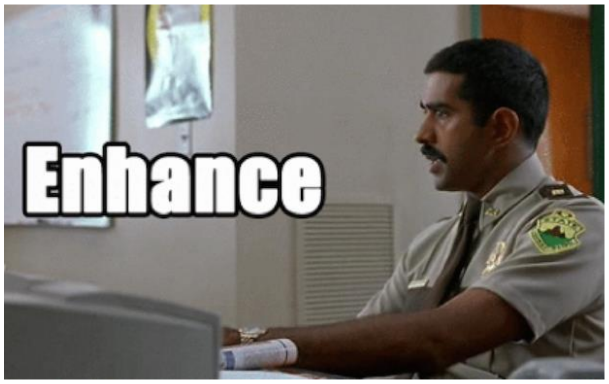
LAB #10: DYNAMIC MEMORY AND CLASSES

DUE DATE: Wednesday May 6th , before Midnight.

DESCRIPTION:

Ok, this is the last lab, for real. Nicely done, you are almost done with the class. Thanks for your patience with what has turned up to be a very eventful semester. For this last lab we will redo lab 8 but now implementing it using Classes and Dynamic Memory.

LAB #8 – ENHANCE!



Introduction:

Remember Lab #8? The one about statistics? Well, you will remember it by the end of this lab. In that lab we used a static array using a constant for the size and we initialized it, populated it, and did some operations with it. We did that with a sequence of functions. This time we are going to implement it as a Class that will let us use dynamic memory to create an array of any size and do the same calculations as before. You already have 60% of the lab done, the functions stay the same, what changes this time is the implementation and management of the data.

START

We are going to create a Class called **Dynamic_Statistics.**

This class will have the following attributes:

Private:

list_size : this is the size of the current dynamic array
numbers_list : this is a dynamic memory array of integer numbers.

Public:

create_array(int n_size):
function of type **bool** that tries to **create a dynamic array using n_size.** If numbers_list already exists (not NULL) then returns **false**. If it can create the dynamic array then n_size is stored in list_size, and returns **true.**

clear_array( ):
function of type bool that tries to free the memory used by the dynamic array. If numbers_list doesn't exist (it is NULL), then it returns **false**. If it does exist, then the memory is freed, list_size is set to 0, and **true** is returned.

initialize_array( ):
void function that checks that the numbers_list has been created and then assigns the value of 0 to all its elements

print_array( ):
void function that prints the numbers_list if it exists. A message should be displayed if the array has not been created.

get_mean( ):
int function that returns the mean of the numbers_list (check that numbers_list has been created).

get_min( ):
int function that returns the minimum value in numbers_list (check that numbers_list has been created).

get_max( ):
int function that returns the maximum value in numbers_list (check that numbers_list has been created)

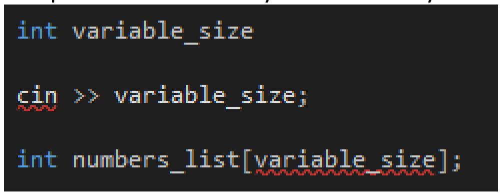
sort_numbers( ):
void function that sorts the list of numbers (check that numbers_list has been created)

get_mean( ): → mean or median? Ask Alex
int function that gets the median value for the list of numbers (check that numbers_list has been created)

display_mode_value( ):
void function that displays the median value (or values if there are more than 1) for the list of numbers (check that numbers_list has been created)

IMPORTANT THINGS TO CONSIDER:

- The point of this lab is dynamic memory. Don't ask for an int variable and then do something like:



```
int variable_size

cin >> variable_size;

int numbers_list[variable_size];
```

  This is not dynamic memory allocation, you need to make sure you use **new and delete.**

- Notice that the functions that are part of the class do not need parameters. Everything is part of the class (encapsulation) so you already have access to it. Make sure you keep public and private sections of your code as defined before.

- There will be no points for implementation that do not use Classes. You can't just resubmit lab8 and get points for it ;)

- Ask questions :D

- Modify the code that was given in lab #8 so that :
  - It uses your class
  - Repeats the code by asking the user to enter a number for the size. When the user enters a number > 2, the code gets executed just like in lab #8.
  - When the user enters any number <= 1 , then the memory that has been used is freed, and the program stops execution.

WHAT YOU NEED TO SUBMIT:

In the dropbox for Lab 10 upload your file for the code and a screenshot of your C++ console program running your code.