Assignment 1

COMP-322, Winter 2017

Due: February 9th, 2016 - 11:55pm

Please read the entire pdf before starting. You must do this assignment individually. There are two parts to this assignment. The first is similar to a quiz and will show you some code and ask you to explain what happens and why. The 2nd involves programming.

Part 1: Written part (40 points)

The purpose of the following questions is to give you some practice with pointers and reference types in C++

The following questions should be answered in a file `Memory.txt`. Your answers should be a few sentences each.

Try to answer these questions first without running the code. You can/should run the code as a way to check your answers, but it's best if you treat it like a take home test and try to figure it out first on your own. Note that this code was tested on the machine linux.cs.mcgill.ca using the g++ compiler and the solutions will be based off of the g++ compiler.

Question 1 (8 points)
Consider the following code: What will print when the program executes and why? What would happen if we changed the method test1 (and the associated prototype) to take as input a reference to an int instead (that is, by inserting an ambersand before the `a` in the method header).

```
#include <iostream>
void test1(int a, int* b);
int main() {
  int x = 3;
  test1(x, &x);

}

void test1(int a, int* b) {
  if (&a == b) {
    std:: cout << "equal!";
  }
  else {
    std:: cout << "not equal";
  }
}
```

Question 2 (8 points)
Consider the following code: What will (most likely) print when the program executes and why?

```
#include <iostream>

int* func1();
```

```
int* func2();

int main() {
  int* a = func1();
  int* b = func2();
  if (a == b) {
    std:: cout << "equal";
  }
  else {
    std :: cout << "not equal";

  }
}

int* func1() {
  int x;
  return &x;
}

int * func2() {
  int x;
  return &x;
}
```

Question 3 (8 points)
Is the following snippet of code good memory management (meaning no segmentation faults or memory leaks)? Why?

```
int * p = new int(5);
int *q = p;
delete p;
delete q;
```

Question 4 (8 points)
Is the following snippet of code good memory management (meaning no segmentation faults or memory leaks)? Why?

```
int * p = new int(5);
int *q = p;
delete p;
p = NULL;
```

Question 5 (8 points)
Consider the following snippet of code that is designed to ask the user to enter their name. Suppose the user types the text `12345678910`. What will print when the value of the variable x is printed and why? (Assume for this question that a char is 1 byte and an int is 4 bytes.)

```
  char name[10];
  int x = 0;
  std:: cout << "Enter your name" << std::endl;
  std:: cin >> name;

  std::cout << x << " ";
```

Part 2 Programming Part (60 points)

The purpose of this question is to give you practice writing a program in C++. The below code should go into a file `RuntimeComparison.cc`*. Remember to compile and run your code using g++ on one of the Linux machines in Trottier before submission.*

First, write 2 methods:

1. `void bubbleSort(int* p, int length)` This method should sort the given array using bubble sort.
2. `void mergeSort(int* p, int length)` This method should sort the given array using merge sort. (Remember that to use recursion you will need to declare the prototype with the function header prior to the method definition.) If you need help with mergesort, please view the wikipedia pages for it or consult the slides and notes at `http://www.cim.mcgill.ca/~langer/250.html` (section number 14).

In both cases assume that the address pointed to by p is large enough to fit `length` ints.

After writing these two functions, you will perform an experiment to compare the two. Write a main function in which you call each of the two functions first on an array of size 10, then 20, then 40, 80, 160, etc until you reach the point that your bubble sort takes more than a few seconds to run. Use the ctime libraries to automatically time your results. See `http://www.cplusplus.com/reference/ctime/clock/` for examples of using this.

You may choose to fill your arrays with random numbers using (for example) what is listed at `http://www.cplusplus.com/reference/cstdlib/rand/`. Note that both of these libraries, as indicated by the library name are copied from the C library, rather than C++. This is done for now, mainly for simplicity. However, you may experiment with using the C++ libraries as well. These can be explored at `http://www.cplusplus.com/reference/`

You may also, if you like, choose to use any of the stl libraries for C++ such as the vector class. You may not use any of the methods in the algorithms library (such as the sort method :)).

Your final program should output a table. Each row of the table should have 3 values in it: the array size `n`, the number of milliseconds for bubble sort, and the number of milliseconds for merge sort. You should output this table with tabs between the values.

Lastly, use a tool such as Open Office, Excel, Google Sheets, etc (whatever you find easiest) to graph a scatter plot of the run times over the size of the array. Export a picture of the graph to a pdf. (Don't worry about the exact format of the graph as far as things like graph title, etc. As long as there is a scatter plot where the y-axis is runtime, the x-axis is input size, and there are dots for both bubble and merge sort.)

You should hand in the source code in a file `RuntimeComparison.cc` as well as the pdf in a file `Runtime.pdf`.

What To Submit

You have to submit one zip file called `Assignment1_YourName.zip` with all your files in it to MyCourses under Assignment 1.
`Memory.txt`
`RuntimeComparison.cc`
`Runtime.pdf`