UNLV UNIVERSITY OF NEVADA, LAS VEGAS

Homework Assignment

Class:	CS202	**Semester:**	Fall 2019
Assignment type:	Homework assignment	**Due date:**	11/10/19
Assignment topic:	Sorting	**Assignment no.**	6
Delivery:	WebCampus – cpp files and txt file Online survey		

Goal

Compare the sorting times of bubble sort and qsort algorithms

Input to the program

Input is internal: code in `main()` function is used to test the functionality

Procedure for the implementation

Develop the sorting program.

General remarks

- Start early as bubble sort will take a longer time
- It's possible to submit the job on bobby and log off, while the job is still running

Steps to develop the program

1. **Write a program, that will:**

   a. define the `Student` class with the following public members in this order:

      i. `int id`

      ii. `double gpa`

   b. outside `main()` function:

      i. write `display()` function, that will display the array as `Student` id:s with gpa:s separated by spaces (no endlines, just one at the end), example: `10034:3.49 16431:3.71` and so on

      ii. write `sortBubble()` function to sort the array by gpa

      iii. write any necessary function(s) required by `qsort` to sort the array by gpa. Don't write your own `qsort`, use the library function.

   c. in `main()` function:

      i. declare array of `Student` objects, size 1 million (use `ARR_SIZE`, as in code skeleton)

      ii. fill `Student` id:s with random values in range `[ID_MIN, ID_MAX)`

      iii. fill `Student` gpa:s with random values in range `[0.0, 4.0]`

      iv. display unsorted list (only during testing for smaller data set)

      v. create user menu: option 1=bubble sort, 2=qsort

      vi. Ask user if to sort using `qsort` or bubble sort

      vii. sort array by `Student` gpa using selected algorithm, measure time using `<ctime>` library functions

      viii. display sorted list (only during testing for smaller data set)

      ix. display sorting time

Use `ARR_SIZE`, `ID_MIN`, `ID_MAX` constants in your code.

Testing

1. Test for smaller set of data

Test your program for proper sorting (both for bubble and **qsort**) using smaller array (e.g. 1000). This is why you have **display()** function. You do the test for the small set to ensure your algorithms work properly.

Executing your program the following way:

```
clear && g++ -Wall -Wextra ./01.cpp -o 01.o && ./01.o < in.txt > out.txt
```

will redirect your output to the **out.txt** file instead of to the screen. It makes it easier to deal with massive screen output. File **in.txt** contains single character: 1 or 2 – depending on the intended menu input.

2. Test for 1M data set

Run your program for 1M array size, don't use **display()** function. Do the sorting for bubbleSort, do the sorting for **qsort**. Fill out the following result table and place it in a txt file.

```
-----------------------------
|  Algorithm  |    Time     |
-----------------------------
| Bubble      | 00h 00m 00s |
-----------------------------
| Qsort       | 00h 00m 00s |
-----------------------------
```

Fill in the online survey: https://unlv.co1.qualtrics.com/jfe/form/SV_1Ung7hYtn6oyrGZ

Submission:

Include the following elements in your submission: (**rid** = your rebel id)

Problem	Element	File
1	Code of your program (for stage 1)	rid_1.cpp file
1	Text file with the table	rid_1.txt file
1	Fill the times in the online survey: https://unlv.co1.qualtrics.com/jfe/form/SV_1Ung7hYtn6oyrGZ	
	Summary of the submission	
	Summary: 1 cpp file, 1 txt file, submit it to the WebCampus. Remember about proper names of the files! Fill the times in the online survey.	

Code skeleton:

```
#include <iostream>

using namespace std;

const int ARR_SIZE=1000000;
const int ID_MIN=10000;
const int ID_MAX=99999;

class Student {

};

void display

}

int main() {
    // create an array of 1M elements

    // fill with random values: id=(ID_MIN-ID_MAX);

    // display unsorted (only for small set phase)

    // ask user whether to sort by bubbleSort or quicksort

    // start measuring time

    // sort by selected algorithm

    // stop measuring time

    // display sorted (only for small set phase)

    // display measured time

}
```