[CENG 315 All Sections] Algorithms

Navigation

- Dashboard
  - Site home
  - Site pages
  - My courses
    - 571 - Computer Engineering
      - [CENG 351 All Sections]
      - CENG 300 All Sections
      - CENG 300 Section 4
      - CENG 315 All Sections
        - Participants
        - Badges
        - Competencies
        - Grades
        - General
        - October 13 - October 19
        - October 20 - October 26
        - October 27 - November 2
          - THE0 Discussion Forum
          - THE0
            - Description
            - Submission view
        - November 3 - November 9
        - November 10 - November 16
        - November 17 - November 23
        - November 24 - November 30
        - December 1 - December 7
        - December 8 - December 14
        - December 15 - December 21
        - December 22 - December 28
        - December 29 - January 4
        - January 5 - January 11
        - January 12 - January 18
        - January 19 - January 25
      - CENG 315 Section 3
      - CENG 331 All Sections
      - CENG 331 Section 2
      - CENG 351 Section 3
    - 651 - Music and Fine Arts
    - 612 - Modern Languages (Persian)
    - 642 - Turkish Language

Description | Submission view

THE0

Available from: Friday, October 29, 2021, 11:59 AM
Due date: Wednesday, November 3, 2021, 11:59 PM
Requested files: the0.cpp, test.cpp (Download)
Maximum upload file size: 1 MiB
Type of work: Individual work

Specifications:

- There is **1 task** to be solved in **12 hours** in this take home exam.
- You will implement your solutions in the **the0.cpp** file.
- You are free to add other functions to **the0.cpp**
- Do **not** change the first line of **the0.cpp**, which is **#include "the0.h"**
- Do **not** change the arguments and return value of the function **insertionSort()** in the file **the0.cpp**
- Do **not** include any other library or write include anywhere in your **the0.cpp** file (not even in comments).

- You are given a test.cpp file to **test** your work on **Odtuclass** or your **locale**. You can and you are encouraged to modify this file to add different test cases.
- If you want to **test** your work and see your outputs you can **compile** your work on your locale as:

```
>g++ test.cpp the0.cpp -Wall -std=c++11 -o test
> ./test
```

- You can test your **the0.cpp** on virtual lab environment. If you click **run**, your function will be compiled and executed with test.cpp. If you click **evaluate**, you will get a feedback for your current work and your work will be **temporarly** graded for **limited** number of inputs.
- The grade you see in lab is **not** your final grade, your code will be reevaluated with more inputs after the exam.

The system have the following limits:

- a maximum execution time of 1 minute
- a 256 MB maximum memory limit
- a stack size of 64 MB for function calls (ie. recursive solutions)

- Each task has a complexity constraint explained in respective sections.
- Solutions with longer running times will not be graded.

- If you are sure that your solution works in the expected complexity constrains but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.
- If you solution is correct, the time and memory limits may be adjusted to accept your solution after the lab. Please send an email if that is the case for you.

```
void insertionSort(int* arr, long &comparison, long & swap, int size);
```

In this exam, you are asked to complete the function definition to sort the given array **arr** with **$ascending** order. Your function should also count the number of **$comparison$** and **$swap$** executed during this sorting proccess (Comparisons are only between the values to be sorted only, not your auxiliary comparisons).

You can use the following pseudocode for the base of your insertionSort implementation:

```
i ← 1
while i < length(A)
    x ← A[i]
    j ← i - 1
    while j >= 0 and A[j] > x
        A[j+1] ← A[j]
        j ← j - 1
    end while
    A[j+1] ← x
    i ← i + 1
end while
```

Constraints:

- Maximum array size is 25000.

Evaluation:

- Since this take home exam is only for testing purposes, you will not be graded with the work you have done.

Example IO:

```
1)
initial array = {9, -2, 3, 15} size=4
sorted array = {-2, 3, 9, 15}, comparison=5, swap=2
2)
initial array = {0, -5, -5, -5, 4, 1} size=6
sorted array = {-5, -5, -5, 0, 1, 4}, comparison=9, swap=4
3)
initial array = {1, 5, 8, 10, 11, 17, 22} size=7
sorted array = {1, 5, 8, 10, 11, 17, 22}, comparison=6, swap=0
```

Requested files

the0.cpp

```
#include "the0.h"

void insertionSort(int* arr, long &comparison, long & swap, int size)
{

    //Your Code Here

}
```

test.cpp

```
//This file is entirely for your test purposes.
//This will not be evaluated, you can change it and experiment with it as you want.
#include <iostream>
#include <fstream>
#include <random>
#include <ctime>
#include "the0.h"

//the0.h only contains declaration of the function insertionSort which is:
//void insertionSort(int* arr, long &comparison, long & swap, int size);

using namespace std;

void randomFill(int*& arr, int size, int minval, int interval)
{
    arr = new int [size];
    for (int i=0; i <size; i++)
    {
        arr[i] = minval + (random() % interval);
    }
}

void print_to_file(int* arr, int size)
{
    ofstream ofile;
    ofile.open("sorted.txt");
    for(int i=0;i<size; i++)
        ofile<<arr[i]<<endl;
}

void read_from_file(int*& arr, int& size)
{

    char addr[]= "input01.txt";
    ifstream infile (addr);

    if (!infile.is_open())
    {
        cout << "File \""<< addr
             << "\" can not be opened. Make sure that this file exists." <<endl;
        return;
    }
    infile >> size;
    arr = new int [size];

    for (int i=0; i<size;i++) {

        infile >> arr[i];
    }

}


void test()
{

    clock_t begin, end;
    double duration;

    //data generation and initialization- you may test with your own data
    long comparison=0;
    long swap=0;
    int size=25000;
    int minval=0;
    int interval=size*10;
    int *arr;

    //Randomly generate initial array:
    //randomFill(arr, size, minval, interval);

    //Read the test inputs. input01.txt through input05.txt exists.
    read_from_file(arr, size);

    //data generation or read end

    if ((begin = clock() ) ==-1)
        cerr << "clock error" << endl;

    //Function call for the solution
    insertionSort(arr, comparison, swap, size);
    //Function end

    if ((end = clock() ) ==-1)
        cerr << "clock error" << endl;

    //Calculate duration and print output

    duration = ((double) end - begin) / CLOCKS_PER_SEC;
    cout << "Duration: " << duration << " seconds." <<endl;
    cout<<"Number of Comparisons: " << comparison <<endl;
    cout<<"Number of Swaps: " << swap <<endl;
    print_to_file(arr,size);
    //Calculation and output end

}

int main()
{
    srandom(time(0));
    test();
    return 0;
}
```

VPL