Dear ODTÜClass Users,
You can access/login ODTUClass 2023-2024 Spring semester at https://odtuclass2023s.metu.
If you want to access the page directly by typing **odtuclass.metu.edu.tr**, **please clear the cache** i
Best regards,
ODTÜClass Support Team

[CENG 315 ALL Sections] Algorithms

Description Subr

THE1

Available from: Saturday, November 4, 2023, 11:59 AM
Due date: Sunday, November 5, 2023, 11:59 PM
Requested files: the1.cpp, test.cpp (Download)
Type of work: Individual work

Problem

In this exam, you are asked to complete the **quickSort()** function definition to sort the given array

```
int quickSort(unsigned short* arr, long &swap, double & avg_dist, double & max_dist, bool hoare, bool median_of_3, int size);
```

You are expected to implement three variants of quickSort() in one function definition as follows:

- ***Quicksort with Lomuto Partitioning*** is called using the function **quickSort()** with **hoare=false** partitioning algorithm in the partition step. You can find the relevant pseudocode below.
- ***Quicksort with Hoare Partitioning*** is called using the function **quickSort()** with **hoare=true**. partitioning algorithm in the partition step. You can find the relevant pseudocode below.
- ***Quicksort with Median of 3 Pivot Selection*** is called using the function **quickSort()** with **med** you should select and arrange a better pivot according to the median of 3 pivot selection algorit two partitioning algorithms. It is a simple algorithm: First, find the median of the first, last, and m

meaning the index floor((size-1)/2)) elements. Then, swap this median with the element in the
partition function. According to the partitioning algorithm, the pivot position may differ. If a swa
variables (**swap**, **avg_dist** etc.). ***Clarification:*** *You are not expected to perform any swap ope*

For all 3 tasks:
You should sort the array in ***descending*** *order, count the number of* ***swap****s executed during the s*
distance between swap positions as ***avg_dist****, find the max distance between swap positions as* ***m***
swap occurs). Finally, the ***quickSort()*** *function should return the number of recursive calls.*

You may notice that there will be swaps in which both sides are pointed by the ***same*** *indexes du*
handle anything. Just like other swaps, apply the swap, increment your swap variable, and updat

For partition tasks follow these pseudocodes exactly:

```
# PSEUDOCODE FOR QUICKSORT WITH CLASSICAL PARTITIONING
PARTITION(arr[0:size-1])

    X←arr[size-1]
    i←-1
    for j←0 to size-2                                  // The last ele
        do if arr[j]≥x
            then i←i+1
                 swap arr[i]↔arr[j]
    swap arr[i+1]↔arr[size-1]
    return i+1

QUICKSORT-CLASSICAL(arr[0:size-1])

    if size>1
        then P←PARTITION(arr[0:size-1])
             QUICKSORT-CLASSICAL(arr[0:P-1])          //P is excluded
             QUICKSORT-CLASSICAL(arr[P+1:size-1])
```

```
# PSEUDOCODE FOR QUICKSORT WITH HOARE PARTITIONING
HOARE(arr[0:size-1])

    X←arr[floor((size-1)/2)]       // i.e. 1 when size=3,4 ---- 2 w
    i←-1
    j←size
    while True
        do  repeat j←j-1
                until arr[j]≥x
            repeat i←i+1
                until arr[i]≤x
            if i<j
                then swap arr[i]↔arr[j]
                else return j

QUICKSORT-HOARE(arr[0:size-1])

    if size>1
        then P←HOARE(arr[0:size-1])
             QUICKSORT-HOARE(arr[0:P])          //P is now include
             QUICKSORT-HOARE(arr[P+1:size-1])
```

Specifications:

- There are 3 **tasks** to be solved in **36 hours** in this take-home exam.
- You will implement your solutions in **the1.cpp** file.
- You are free to add other functions to **the1.cpp**
- Do **not** change the first line of **the1.cpp**, which is **#include "the1.h"**
- Do **not** change the arguments and return value of the functions **quickSort()** in the file **the1.cpp**
- Do **not** include any other library or write include anywhere in your **the1.cpp** file (not even in com

- You are given **test.cpp** file to **test** your work on **ODTUClass** or your **locale**. You can (and you ar 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 the1.cpp -Wall -std=c++11 -o test

> ./test
```

- You can test your **the1.cpp** on the virtual lab environment. If you click **run**, your function will be with **test.cpp**. If you click **evaluate**, you will get feedback for your current work and your work v a **limited** number of inputs.
- The grade you see in lab is **not** your final grade, your code will be reevaluated with **completely**

The system has the following limits:

- a maximum execution time of 32 seconds (your functions should return in less than 1 seconds f
- a 192 MB maximum memory limit
- an execution file size of 1M.
- Solutions with longer running times will not be graded.

- If you are sure that your solution works in the expected complexity constraints but your evaluati environment, the constant factors may be the problem.

Evaluation:

- After your exam, black box evaluation will be carried out. You will get full points if you set all the

Example IO:

```
1)
initial array = {4, 3, 2, 1}, size=4
sorted array = {4, 3, 2, 1}
Classical Lomuto partitioning -> swap=9, avg_dist=0, max_dist=0, n_calls=7
Classical Hoare Partitioning -> swap=0, avg_dist=0, max_dist=0, n_calls=7
2)
initial array = {1, 2, 3, 4} size=4
sorted array = {4, 3, 2, 1}
Classical Lomuto partitioning -> swap=5, avg_dist=0.8, max_dist=3, n_calls=7
Classical Hoare partitioning -> swap=2, avg_dist=2, max_dist=3, n_calls=7
Median of 3 Lomuto partitioning -> swap=6, avg_dist=0.833333, max_dist=2, n_calls=5
Median of 3 Hoare partitioning -> swap=2, avg_dist=2, max_dist=3, n_calls=7
3)
initial array = {5, 23, 3, 98, 45, 1, 90}, size=7
sorted array = {98, 90, 45, 23, 5, 3, 1}
Classical Lomuto partitioning -> swap=6, avg_dist=2.66667, max_dist=5, n_calls=9
Classical Hoare partitioning -> swap=6, avg_dist=1.83333, max_dist=4, n_calls=13
Median of 3 Lomuto partitioning -> swap=7, avg_dist=2.28571, max_dist=5, n_calls=7
Median of 3 Hoare partitioning -> swap=6, avg_dist=3, max_dist=6, n_calls=13
```

Requested files

the1.cpp

```cpp
#include "the1.h"

//You may write your own helper functions here

int quickSort(unsigned short* arr, long& swap, double& avg_dist, double& max_dist, boo
    //Your code here

}

```

test.cpp

```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 "the1.h"

using namespace std;

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

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

void read_from_file(unsigned short*& input_array, int& size, bool& hoare, bool& media

    char addr[]= "inp01.txt";
    ifstream infile (addr);
    if (!infile.is_open())
    {
        cout << "File \'"<< addr
            << "\' can not be opened. Make sure that this file exists." <<endl;
        return;
    }
    infile >> hoare;
    infile >> median_of_3;
    infile >> size;
    input_array = new unsigned short[size];
    for(int j=0; j<size; j++){
        infile >> input_array[j];
    }

}


void test(){
    int size = 1 << 8;
    int number_of_recursive_calls;
    long swap=0;
    double avg_dist=0, max_dist=0;
    bool hoare=true, median_of_3=true;
    bool rand_fill = false;
    unsigned short* input_array;
    unsigned short minval=0;
    unsigned short interval= (unsigned short)((1<<16)-1);   // unsigned short 65535 i


    if(rand_fill)
        randomFill(input_array, size, minval, interval);    //Randomly generate initi
    else
        read_from_file(input_array, size, hoare, median_of_3);

```