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

THE2

Available from: Saturday, November 11, 2023, 12:00 PM
Due date: Sunday, November 12, 2023, 11:59 PM
Requested files: the2.cpp, test.cpp (Download)
Type of work: Individual work

Your librarian friend is planning to digitize the catalog of the local library and needs your help. He the online catalog and asks you to sort them in a dictionary-like order. You decide to help him by Radix Sort algorithm to sort the given strings.

Problem

In this exam, you are asked to sort the given string array ***arr*** with Radix Sort in ascending or desce boolean variable ***ascending*** and return the number of iterations done in the loops of the Counting Counting Sort as a subroutine in the Radix Sort). ***size*** is the number of elements in the string array

```
int radix_string_sort(std::string *arr, int size, bool ascending);
```

Constraints and Hints:

- Array elements will be strings each of which can contain only the characters as **uppercase Eng**

- It will be easier to follow the iteration count if you implement your solution by modifying the pse
- Different from the Radix Sort algorithm in your book, it is **not guaranteed** that the strings in the length. (***Hint:*** You can use an extra bucket during Counting Sort to handle strings with different
- Different than the algorithm for Counting Sort in your book, initialize the count array as *int* C =* copying the array back. That means, you shouldn't count iterations during initialization, but you copying array back. Otherwise, the return value of the function (as the number of iterations) will
- You should count loop iterations in four different loops.
- *Ascending order means dictionary order*. For example, when **ascending=true**, OGUZ must resi OGUZHAN. If **ascending=false**, it is the other way around.

Evaluation:

- After your exam, black-box evaluation will be carried out. You will get full points if you fill the ***arr*** number of iterations correctly for the cases that will be tested.

Example IO:

1) Size: 5, Ascending: True

Array elements: {OGUZ, UMAY, ASLI, EMRE, EREN}

Number of iterations: 164

Sorted array: {ASLI, EMRE, EREN, OGUZ, UMAY}

2) Size: 6, Ascending: False

Array elements: {PAPATYA, LALE, MENEKSE, AKSAMSEFASI, BEGONVIL, KARANFIL}

Number of iterations: 484

Sorted array: {PAPATYA, MENEKSE, LALE, KARANFIL, BEGONVIL, AKSAMSEFASI}

3) Size: 7, Ascending: True

Array elements: {BETELGEUSE, VEGA, SEGINUS, SEGIN, ANTARES, ALDERAMIN, ALDEBERAN}

Number of iterations: 470

Sorted array: {ALDEBERAN, ALDERAMIN, ANTARES, BETELGEUSE, SEGIN, SEGINUS, VEGA}

4)Size: 7, Ascending: False

Array elements: {BETELGEUSE, VEGA, SEGINUS, SEGIN, ANTARES, ALDERAMIN, ALDEBERAN}

Number of iterations: 470

Sorted array: {VEGA, SEGINUS, SEGIN, BETELGEUSE, ANTARES, ALDERAMIN, ALDEBERAN}

Specifications:

- There is 1 task to be solved in **36 hours** in this take-home exam.
- You will implement your solutions in ***the2.cpp*** file.
- You are free to add other functions to *the2.cpp*
- **Do not change** the first line of the2.cpp, which is #include "the2.h"
- *<string> is* included in "the2.h" for your convenience.

- **Do not change** the arguments and the return value of the function ***radix_string_sort()*** in the fil
- **Do not include** any other library or write include anywhere in your the2.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 the2.cpp -Wall -std=c++11 -o test

> ./test
```

- You can test your the2.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 a limited number of inputs.
- The grade you see in lab is not your final grade, **your code will be reevaluated with different i**

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, but your evaluation fails due constant factors may be the problem.

Requested files

the2.cpp

```cpp
#include "the2.h"

// do not add extra libraries here
/*
    arr       : array to be sorted, in order to get points this array should contain
    ascending : true for ascending, false for descending
    size      : number of elements in the array
*/
int radix_string_sort(std::string* arr, int size, bool ascending){

    return 0;

}

```

test.cpp

```cpp
// this file is for you for testing purposes, it won't be included in evaluation.

#include <iostream>
#include <fstream>
#include "the2.h"

void file_input(std::string*& input_array, int& size, bool& ascending){
    std::string file_name = "inp06.txt"; // inp01-inp10 are available.
    std::ifstream infile (file_name);
    if(!infile.is_open()){
        std::cout << "Input file cannot be opened" << std::endl;
        std::cout << "File name: " << file_name << std::endl;
        return;
    }
    infile >> ascending;
    infile >> size;
    input_array = new std::string[size];
    for(int j=0; j<size; j++){
        infile >> input_array[j];
    }
    return;
}

void test(){
    int number_of_iteration, size;
    bool ascending;
    std::string* input_array;
    file_input(input_array, size, ascending);
    std::cout << "Size: " << size << std::endl <<
                 "Ascending: " << ascending << std::endl <<
                 "Array elements: {";
    for(int idx=0; idx < size - 1; idx++) std::cout << input_array[idx] << ", ";
    std::cout << input_array[size-1] << "}" << std::endl;
    number_of_iteration = radix_string_sort(input_array, size, ascending);
    std::cout << "Number of iterations: " << number_of_iteration << std::endl <<
                 "Sorted array: {";
    for(int idx=0; idx<size-1; idx++) std::cout << input_array[idx] << ", ";
    std::cout << input_array[size-1] << "}" << std::endl;
    return;
}

int main(){
    test();
    return 0;
}

```

- 2021-2022 Summer
- 2021-2022 Spring
- 2021-2022 Fall
- 2020-2021 Summer
- 2020-2021 Spring
- 2020-2021 Fall

Class Archive

Get the mobile app