You can view this report online at : https://www.hackerrank.com/x/tests/1696214/candidates/55842019/report

Full Name:	Breeha Qasim
Email:	bq08283@st.habib.edu.pk
Test Name:	**CS224 Lab# 04 - T5 - Fall 2023**
Taken On:	13 Sep 2023 12:13:07 PKT
Time Taken:	8863 min 13 sec/ 7920 min
Work Experience:	< 1 years
Invited by:	Shafaq Fatima
Skills Score:	
Tags Score:	C++ 100/100 Easy 74/74 Functions 174/174 Input 74/74 Pointers 100/100

100%

324/324

scored in **CS224 Lab# 04 - T5 - Fall 2023** in 8863 min 13 sec on 13 Sep 2023 12:13:07 PKT

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here - https://www.hackerrank.com/x/tests/1696214/candidates/55842019/report

	Question Description	Time Taken	Score	Status
Q1	Time 24 Hours > Coding	1 hour 4 min 22 sec	50/ 50	✓
Q2	Complex Calculator - overloaded > Coding	18 min 11 sec	50/ 50	!
Q3	Holes in a Number/Alphabet -- Overloading + Recursion > Coding	37 min 44 sec	74/ 74	✓
Q4	Call Me > Coding	18 min 59 sec	10/ 10	!
Q5	Dynamically allocating/deallocating arrays > Coding	3 hour 38 min 7 sec	20/ 20	✓
Q6	Dynamic char array using char * > Coding	33 min 40 sec	20/ 20	!
Q7	Sorting an array of strings using pointers > Coding	20 min 54 sec	100/ 100	!

QUESTION 1

Correct Answer

Score 50

Time 24 Hours › Coding

QUESTION DESCRIPTION

Write a function validatedTime(hours, minutes, seconds), that prints a valid time in 24 hours format. It adjusts any of the variable if it's out of its valid range e.g. 65 seconds will be adjusted to 5, incrementing 1 in minutes.

main function is provided, you have to make it working.

Function Description

validatedTime have default value of zero for hours, minutes, and second. Hence it can be called by providing none, one, two, or three arguments.

▶ Details
▶ Details

INTERVIEWER GUIDELINES

```
#include<iostream>

using namespace std;

void validatedTime(int hours = 0, int minutes = 0, int seconds = 0){
    if(seconds >= 60){
        minutes += seconds/60;
        seconds %= 60;
    }

    if(minutes >= 60){
        hours += minutes/60;
        minutes %= 60;
    }

    if(hours >= 24){
        hours %= 24;
    }

    if (hours<10) cout<<0;
    cout<<hours<<":";
    if (minutes<10) cout<<0;
    cout<<minutes<<":";
    if (seconds<10) cout<<0;
    cout<<seconds<<endl;
}


int main(){
    int hours, minutes, seconds;
    cin>>hours>>minutes>>seconds;
    validatedTime();
    validatedTime(hours);
    validatedTime(hours, minutes);
    validatedTime(hours, minutes, seconds);
}
```

```
#include<iostream>

using namespace std;
class Time24{
    int hours, minutes, seconds;
    public:
    Time24(int h, int m, int s): hours(h), minutes(m), seconds(s){
        validatedTime();
    }
```

```
    Time24(int h, int m, int s, char p): hours(h), minutes(m), seconds(s)
{
        if (p=='p') hours+=12;
        validatedTime();
    }
    void add(Time24 t){
        hours+= t.hours;
        minutes+= t.minutes;
        seconds+= t.seconds;
        validatedTime();
    }
void validatedTime(){
    if(seconds >= 60){
        minutes += seconds/60;
        seconds %= 60;
    }

    if(minutes >= 60){
        hours += minutes/60;
        minutes %= 60;
    }

    if(hours >= 24){
        hours %= 24;
    }
}
void show() const{
    if (hours<10) cout<<0;
    cout<<hours<<":";
    if (minutes<10) cout<<0;
    cout<<minutes<<":";
    if (seconds<10) cout<<0;
    cout<<seconds<<endl;
}
};

int main(){
    int hours1, minutes1, seconds1;
    cin>>hours1>>minutes1>>seconds1;
    Time24 t1(hours1, minutes1, seconds1);


    int hours2, minutes2, seconds2;
    char period;
    cin>>hours2>>minutes2>>seconds2>>period;
    const Time24 t2 = {hours2, minutes2, seconds2, period};

    cout<<"t1: "; t1.show();
    cout<<"t2: "; t2.show();

    t1.add(t2);
    cout<<"t1+t2: "; t1.show();
}
```

CANDIDATE ANSWER

Language used: **C++**

```
#include<iostream>
using namespace std;

void validatedTime(){
    cout<<"00:00:00"<<endl;
}

```

```
void validatedTime(int hours){
    while (hours>24){
        hours=hours-24;
    }
    std::string hour = std::to_string(hours);
    if (hour.size()==1){
        hour= '0'+hour;
    }
    cout<<hour<<":00:00"<<endl;
}

void validatedTime(int hours,int minutes){
    while (minutes>=60){
        hours++;
        minutes=minutes-60;
        while (hours>24){
            hours=hours-24;
        }
    }
    while (hours>24){
        hours=hours-24;
    }

    std::string hour = std::to_string(hours);
    if (hour.size()==1){
        hour= '0'+hour;
    }

    std::string minute = std::to_string(minutes);
    if (minute.size()==1){
        minute= '0'+minute;
    }
    cout<<hour<<":"<<minute<<":00"<<endl;
}

void validatedTime(int hours, int minutes, int seconds){
    while (seconds>=60){
        minutes++;
        seconds=seconds-60;
        while (minutes>=60){
            hours++;
            minutes=minutes-60;
            while (hours>24){
               hours=hours-24;
            }
        }
    }
    while (minutes>=60){
            hours++;
            minutes=minutes-60;
            while (hours>24){
               hours=hours-24;
            }
    }
    while (hours>24){
            hours=hours-24;
    }

    std::string hour = std::to_string(hours);
    if (hour.size()==1){
        hour= '0'+hour;
    }

```

```
    std::string minute = std::to_string(minutes);
    if (minute.size()==1){
        minute= '0'+minute;
    }

    std::string second = std::to_string(seconds);
    if (second.size()==1){
        second= '0'+second;
    }
    cout<<hour<<":"<<minute<<":"<<second<<endl;

}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.052 sec	9.01 KB
Testcase 1	Easy	Sample case	Success	10	0.0206 sec	9.01 KB
Testcase 2	Easy	Sample case	Success	10	0.0281 sec	8.92 KB
Testcase 3	Easy	Hidden case	Success	10	0.0222 sec	8.91 KB
Testcase 4	Easy	Hidden case	Success	10	0.0304 sec	8.8 KB

No Comments

QUESTION 2

Needs Review

Score 50

Complex Calculator - overloaded › Coding

QUESTION DESCRIPTION

You have to define functions to add, subtract, and multiply two complex numbers. Since complex numbers can be added/multiplied/subtracted with a real number as well, so the add, subtract and multiply functions should be overloaded: one where both operands are Complex numbers, other where first operand is complex and second operand is real.
A function show will display the complex number is usual way i.e. x+yi

The function needs to return a complex number in the representation of an array.

You have to define the show function yourself.

Main function is already given, you have to make it working.

Note: We have represented a complex number using an array of 2 elements of type double, where the first element represents real part and the second one represent imagenary part of the complex number. so you have to follow the same convention to represent complex number.

Refer to this page for multiplication of complex numbers:
https://www.mathsisfun.com/algebra/complex-number-multiply.html

▶ Details

INTERVIEWER GUIDELINES

```
#include<iostream>
using namespace std;


double* add(double* c1, double* c2){
    double *addition = new double[2];
    addition[0] = c1[0]+c2[0];
    addition[1] = c1[1]+c2[1];
```

```
    return addition;
}

double * add(double* c1, double d1){
    double *add2 = new double[2];
    add2[0] = c1[0] + d1;
    add2[1] = c1[1];

    return add2;
}

double * subtract(double* c1, double* c2){
    double *sub = new double[2];
    sub[0] = c1[0]-c2[0];
    sub[1] = c1[1]-c2[1];
    return sub;
}

double* subtract(double* c1, double d1){
    double *sub2 = new double[2];
    sub2[0] = c1[0] - d1;
    sub2[1] = c1[1];

    return sub2;
}

double* multiply(double* c1, double* c2){
    double *mul = new double[2];
    mul[0] = c1[0]*c2[0]-c1[1]*c2[1];
    mul[1] = c1[0]*c2[1]+c1[1]*c2[0];

    return mul;
}

double* multiply(double* c1, double d1){

    double *mul = new double[2];
    mul[0] = c1[0] * d1;
    mul[1] = c1[1] * d1;

    return mul;
}
void show(double* c){
    if(c[1]>=0)
    cout<<c[0]<<"+"<<c[1]<<"i";
    else
    {
    cout<<c[0]<<c[1]<<"i";
    }
    cout<<endl;
}


int main()
{
    double * c1 = new double[2]; // there are two elements in this array
the first one represents the real part and second one represents
imagenary part
    double * c2 = new double[2];

    cin>>c1[0]>>c1[1];
    cin>>c2[0]>>c2[1];

    double d1;
    cin>>d1;

    cout<<"c1+c2: "; show(add(c1,c2));
    cout<<"c1-c2: "; show(subtract(c1,c2));
    cout<<"c1*c2: "; show(multiply(c1,c2));
```

```
    cout<<"c1+d1: "; show(add(c1,d1));
    cout<<"c1-d1: "; show(subtract(c1,d1));
    cout<<"c1*d1: "; show(multiply(c1,d1));
}
```

```
#include<iostream>

using namespace std;

struct Complex{
    double real, imag;
};

Complex add(Complex c1, Complex c2){
    return {c1.real+c2.real, c1.imag+c2.imag};
}

Complex add(Complex c1, double d1){
    return {c1.real + d1, c1.imag};
}

Complex subtract(Complex c1, Complex c2){
    return {c1.real-c2.real, c1.imag-c2.imag};
}

Complex subtract(Complex c1, double d1){
    return {c1.real-d1, c1.imag};
}

Complex multiply(Complex c1, Complex c2){
    return {c1.real*c2.real-c1.imag*c2.imag,
c1.real*c2.imag+c1.imag*c2.real};
}

Complex multiply(Complex c1, double d1){
    return {c1.real*d1, c1.imag*d1};
}
void show(Complex c){
    if(c.imag>0)
    cout<<c.real<<"+"<<c.imag<<"i";
    else
    {
    cout<<c.real<<c.imag<<"i";
    }
    cout<<endl;
}


int main(){
    Complex c1, c2;
    cin>>c1.real>>c1.imag;
    cin>>c2.real>>c2.imag;

    double d1;
    cin>>d1;

    cout<<"c1: "; show(c1);
    cout<<"c2: "; show(c2);
    cout<<"d1: "<<d1<<endl;

    cout<<"c1+c2: "; show(add(c1,c2));
    cout<<"c1-c2: "; show(subtract(c1,c2));
    cout<<"c1*c2: "; show(multiply(c1,c2));


    cout<<"c1+d1: "; show(add(c1,d1));
    cout<<"c1-d1: "; show(subtract(c1,d1));
```

```
    cout<<"c1*d1: "; show(multiply(c1,d1));
}
```

CANDIDATE ANSWER

Language used: **C++**

```
1
2  // Avoid the use of "using namespace std;"
3  // Expand all provided code to understand what is going on.
4  double* add(double* c1, double* c2)
5  {
6      double* result = new double[2];
7      result[0] = c1[0] + c2[0];
8      *(result+0) = *(c1+0) + *(c2+0);
9      *(result+1) = *(c1+1) + *(c2+1);
10     return result;
11 }
12 double* add(double* c1, double var)
13 {
14     double* result = new double[2];
15     *result = *c1 + var;
16     *(result+1) = *(c1+1);
17     return result;
18 }
19 double* subtract(double* c1, double* c2)
20 {
21     double* result = new double[2];
22     result[0] = c1[0] - c2[0];
23     result[1] = c1[1] - c2[1];
24     return result;
25 }
26 double* subtract(double* c1, double var)
27 {
28     double* result = new double[2];
29     result[0] = c1[0] - var;
30     result[1] = c1[1];
31     return result;
32 }
33 double* multiply(double* c1, double* c2)
34 {
35     double* result = new double[2];
36     result[0] = (c1[0]*c2[0])-(c1[1]*c2[1]);
37     result[1] = (c1[0]*c2[1])+(c1[1]*c2[0]);
38     return result;
39 }
40 double* multiply(double* c1, double var)
41 {
42     double* result = new double[2];
43     result[0] = (c1[0]*var);
44     result[1] = (c1[1]*var);
45     return result;
46 }
47 void show(double* result)
48 {
49     if(result[1] <= 0)
50     {
51        std::cout << result[0] << result[1] << "i"<<"\n";
52     }
53     else
```

```
    {
    std::cout << result[0] << "+" << result[1] << "i"<<"\n";
    }
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.1192 sec	8.89 KB
Testcase 1	Easy	Sample case	Success	10	0.066 sec	8.93 KB
Testcase 2	Easy	Sample case	Success	10	0.0257 sec	8.93 KB
Testcase 3	Easy	Hidden case	Success	10	0.0562 sec	9.08 KB
Testcase 5	Easy	Hidden case	Success	10	0.0559 sec	8.99 KB

No Comments

QUESTION 3

Correct Answer

Score 74

Holes in a Number/Alphabet -- Overloading + Recursion › Coding

Easy | Functions | Input

QUESTION DESCRIPTION

You are designing a poster which prints out numbers with a different style applied to each of them. The styling is based on the number of closed paths or *holes* present in the numbers and alphabets.

The number of holes present in each of the digits from 0 to 9 is equal to the number of closed paths in the digit. Their values are:

- 1, 2, 3, 5, and 7 = 0 holes.
- 0, 4, 6, and 9 = 1 hole.
- 8 = 2 holes.
- C,E,F,G,H,I,J,K,L,M,N,S,T,U,V,W,X,Y,Z = 0 hole
- A,D,O,P,Q,R = 1 hole
- B = 2 hole

The total number of holes in the number, 1078, is 3 which is the the sum of

- the number of holes in 1 = 0 holes.
- the number of holes in 0 = 1 hole.
- the number of holes in 7 = 0 holes.
- the number of holes in 8 = 2 holes.

Similarly an example of English alphabets in word "PAKISTAN" would be:

- P = 1, A=1, K=0, I=0, S=0, T=0, A=1, N=0 => Total Holes = 3

Functions Description

Write a recursive function `countHoles` that takes a parameter, `num of type int`, and returns the sum of the number of holes in all of its digits.

Write an overloaded recursive version of **countHoles** that takes a parameter **str of type char *** and returns the sum of the number of holes in all of its alphabets.

Note: You should not use loop to calculate the number of hole, recursion must be used.

Also: I can call the same function multiple times, your function should be able to remember the previous count and return the sum.

e.g if I call countHole with 346, it should return 2, but if I call it again with 789, it should return 5, the previous sum should be added to the current one.

Also: The alphabets will be only in Capitals

▼ Input Format For Custom Testing

The main() is already written where we have handled the input for actual program.

For custum testing the input will be just an int number or a string in capital letters.

▼ Sample Case 0

Sample Input For Custom Testing

```
1078
```

Sample Output

```
3 holes
```

Explanation

The number of holes in 1, 0, 7, and 8 are 0, 1, 0, and 2 respectively which add up to 3.

▼ Sample Case 1

Sample Input For Custom Testing

```
KARACHI
```

Sample Output

```
3 holes
```

Explanation

The number of holes in A=1, R=1 and A=1 which add up to 3

Sample Case 2

Sample Input For Custom Testing

```
1078
6789
```

Sample Output

```
7 holes
```

Explanation

The number of holes in 1, 0, 7, and 8 are 0, 1, 0, and 2 respectively which add up to 3. and then the holes in 6,7,8,9 are 1,0,2,1 respectively which adds up to 4. So the total sum becomes 7.

INTERVIEWER GUIDELINES

Solution

```
#include<string>
#include<cstring>

using namespace std;

int holes[] = {1,0,0,0,1,0,1,0,2,1};

int countHoles(int num)
{
    int count=0;
    if(num > 0)
    {
        count = count + holes[num%10];
```

```
        return count + countHoles(num/10);
    }
    else
        return 0;


}

int countHoles(char *arr)
{
    static int count=0;
    if(*arr!=0)
    {

        if(*arr == 'B')
            count+=2;

        else if(*arr == 'A' ||*arr == 'D' ||*arr == 'O' ||*arr == 'P'
||*arr == 'Q' ||*arr == 'R')
            count+=1;


        return countHoles(arr+1);
    }
    else
        return count;
}


int main()
{
    string input;

    int nHoles=0;

    while(getline(cin,input))
    {
        if(isdigit(input[0]))
        {
            nHoles=countHoles(stoi(input));
        }
        else
        {
            char * carr = new char[input.length()];
            carr = strcpy(carr, input.c_str());
            nHoles =countHoles(carr);
        }
    }

    cout << nHoles <<" holes";
    return 0;
}
```

CANDIDATE ANSWER

Language used: **C++**

```
// write your code here
int countHoles(int num){
    int static holes=0;
    //cout<<num;
    if (num==0){
        return holes;
    }
    else{
```

```
        if (num%10==0 || num%10==4 || num%10==6 || num%10==9){
            holes++;
        }
        else if (num%10==8){
            holes+=2;
        }
    }
    return countHoles(num/10);
}

int countHoles(char* chr){
    int static i=0;
    int static holes=0;
    if (chr[i]=='\0'){
        i=0;
        return holes;
    }
    else if (chr[i]=='A' || chr[i]=='D' ||chr[i]=='O' || chr[i]=='P'||
chr[i]=='Q'  || chr[i]=='R'){
        holes++;
    }
    else if (chr[i]=='B'){
        holes+=2;
    }
    i++;
    return countHoles(chr);
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	2	0.0757 sec	8.86 KB
Testcase 1	Easy	Sample case	Success	2	0.0362 sec	8.93 KB
Testcase 2	Easy	Hidden case	Success	10	0.0596 sec	8.79 KB
Testcase 3	Easy	Hidden case	Success	10	0.0506 sec	8.72 KB
Testcase 4	Easy	Hidden case	Success	10	0.0382 sec	8.8 KB
Testcase 5	Easy	Hidden case	Success	10	0.0655 sec	9.02 KB
Testcase 6	Easy	Sample case	Success	10	0.1157 sec	8.86 KB
Testcase 7	Easy	Hidden case	Success	10	0.0449 sec	8.77 KB
Testcase 8	Easy	Hidden case	Success	10	0.0416 sec	8.81 KB

No Comments

QUESTION 4

Needs Review

Score 10

Call Me › Coding

QUESTION DESCRIPTION

Write a function callMe(), *notice it has no argument,* that, when you call it, displays a message telling how many times it has been called: “I have been called 3 times”, for instance. Write a main() program that calls this function 10 times. Try implementing this function in two different ways:
First, use a global variable to store the count. Second, use a local **static** variable. Which is more appropriate? (the answer is local static variable) Why can’t you use a local variable?

Note: In final solution comment out the first method i.e. using global variable.

INTERVIEWER GUIDELINES

```
#include<iostream>

using namespace std;

void callMe(){
    static int n=0;
    cout<<"I have been called "<<n++<<" times"<<endl;
}
int main(){
    for(int i=0;i<10;i++)
        callMe();
}
```

CANDIDATE ANSWER

Language used: **C++**

```
#include <iostream>
/*/
int callme{1};
void callMe(){
    std::cout<<"I have been called "<< callme++ << " times"<< std::endl;
}
int main(){
    //int callme{1};
    for (int i=0;i<10;i++){
        callMe();
    }
}
/*/

void callMe(){
    static int callme{1};
    std::cout<<"I have been called "<< callme++ << " times"<< std::endl;
}

int main(){
    for (int i=0;i<10;i++){
        callMe();
    }

}

//Static is more appropriate :)
//We can't use local variable because the value will get reset again and
again, using static the value updated will be stored in it
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0447 sec	8.72 KB

No Comments

QUESTION 5

Dynamically allocating/deallocating arrays › Coding

Correct Answer

Score 20

QUESTION DESCRIPTION

In addition to dynamically allocating single values, we can also dynamically allocate arrays of variables. Unlike a fixed array, where the array size must be fixed at compile time, dynamically allocating an array allows us to choose an array length at runtime (meaning our length does not need to be constexpr).

To allocate an array dynamically, we use the array form of new and delete (often called new[] and delete[]).

```
int size;
cin >> size ;

int* num ;
num = new int[size]; // use array new to allocate the array .  Note that
size does not need to be constant!


delete[] num; // use array delete to deallocate the array
```

CANDIDATE ANSWER

Language used: **C++14**

```
#include <iostream>

using namespace std;

int main()
{
    size_t length{};
    cin >> length; // user input for the length of the array

    int* num;  // allocate an array 'num' using new operator of size 'length'
    int *num_arr= new int[length];


    for(int i = 0 ; i < length ; i++){
        // take user input to initialise element of array 'num'
        cin>>num_arr[i];
        cout<<"Element "<<i+1<<": "<< num_arr[i]<<"\n";
        // wriet a loop to print the array content.
    }


     // use array delete to deallocate array

    // we don't need to set array to nullptr/0 here because it's going out of
scope immediately after this anyway

    return 0;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0491 sec	8.89 KB
Testcase 1	Easy	Sample case	Success	10	0.0259 sec	8.66 KB

No Comments

QUESTION 6

Needs Review

Score 20

Dynamic char array using char * › Coding

QUESTION DESCRIPTION

Please fix the code given to pass the given test cases.

CANDIDATE ANSWER

Language used: **C++14**

```
#include <iostream>

using namespace std;

int main()
{

    int length;
    cin >> length; // user input for the length of the array

    char* name = new char[length]; // allocate an array 'name' using new
operator of size 'length'

    for (int i=0; i<length-1; i++){
        char temp=' ';
        cin>>temp;
        if(isupper(temp) and i!=0){
            name[i]=' ';
            i++;
            name[i]=temp;
        name[i]=temp; // user input in name array
        }
        else{
           name[i]=temp; // user input in name array
        }
    }
     cout << name << endl;
     delete [] name;

     // use array delete to deallocate array

    // we don't need to set array to nullptr/0 here because it's going out of
scope immediately after this anyway

    return 0;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0527 sec	8.91 KB
Testcase 1	Easy	Sample case	Success	10	0.0646 sec	8.88 KB

No Comments

QUESTION 7

Needs Review

Score 100

Sorting an array of strings using pointers › Coding

C++ | Pointers | Functions

QUESTION DESCRIPTION

In this program you have to sort an array of strings(char arrays) using pointers.

You have to define the following functions:
1. main function: in this function you will first take n as user input then create an array of size n which can store strings.
   Then take n strings from user and put them into the array of strings. Then pass this array of strings to the sort function. Finally print the strings.
   Hint: You will have to use cin.ignore() while taking input. Learn more about it from https://www.javatpoint.com/cin-ignore-function-in-cpp
and https://cplusplus.com/reference/istream/istream/ignore/
2. sort function: this function take an array of strings and size n as parameters. loop throgh the array and sort all the strings.
   Hint: First you can try strcmp() to compare char arrays, then try to do it without strcmp().

Example input:

```
5
Turkmenistan
Uzbekistan
Jamaica
South Africa
Guyana
```

Example output:

```
Guyana
Jamaica
South Africa
Turkmenistan
Uzbekistan
```

INTERVIEWER GUIDELINES

```
/// define sort() function here

void sort(char ** names, int size)
{
   for (int i = 0; i < size; i++)
   {
      for (int j  = 0; j < size -i -1; j++)
      {
         if(strcmp(names[j], names[j+1]) > 0)
         {
            char* temp = names[j];
            names[j] = names[j+1];
            names[j+1] = temp;
         }
      }

   }

}
```

```
int main() {
    // take the input here and call the function 34567

    int size;
    cin >> size;
    cin.ignore();
    char ** names = new char*[size];
    for (int i = 0; i < size; i++)
    {
        char *c = new char[50];
        cin.getline( c,50);
        //cout << c << endl;
        names[i] = c;
    }


    sort(names, size);
    for (int i = 0; i < size; i++)
    {
        cout << names[i] << endl;
    }

    return 0;
}
```

```
void sort(char** names, int size)
{
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size - i - 1; j++)
        {
            if (strcmp(names[j], names[j+1]) > 0)
            {
                char* temp = names[j];
                names[j] = names[j+1];
                names[j+1] = temp;
            }
        }
    }
}

int main() {
    int size;
    cin >> size;
    cin.ignore();
    char** names = new char*[size];
    for (int i = 0; i < size; i ++)
    {
        char *c = new char[50];
        cin.getline(c, 50);
        names[i] = c;
    }

    sort(names, size);

    for (int i = 0 ; i < size; i++)
    {
        cout << names[i] << endl;
    }

    return 0;

}
```

CANDIDATE ANSWER

Language used: **C++**

```
/// define sort() function here
void sort_function(char* str[],int n)
{
    for (int i=0;i<n;i++){
        for (int j=0;j<n-1-i;j++){
            if (strcmp(str[j],str[j+1])>0){
                char* temp=str[j];
                str[j]=str[j+1];
                str[j+1]=temp;
            }
        }
    }
}



int main()
{
    // write your main() code here
    int n;
    cin>>n;
    cin.ignore();
    char* arr1[n];
    for (int i=0;i<n;i++){
        char strings[10000];
        cin.getline(strings,sizeof(strings));
        arr1[i]= new char[strlen(strings)+1];
        strcpy(arr1[i],strings);
    }
    sort_function(arr1,n);
    for (int i=0;i<n;i++){
        cout<<arr1[i]<<endl;
        delete[] arr1[i];
    }
    return 0;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0329 sec	8.75 KB
Testcase 1	Easy	Hidden case	Success	10	0.0594 sec	8.86 KB
Testcase 2	Easy	Hidden case	Success	10	0.0551 sec	8.95 KB
Testcase 3	Easy	Hidden case	Success	10	0.0522 sec	8.8 KB
Testcase 4	Easy	Hidden case	Success	10	0.0797 sec	8.88 KB
Testcase 5	Easy	Hidden case	Success	10	0.034 sec	8.87 KB
Testcase 6	Easy	Hidden case	Success	10	0.0272 sec	8.86 KB
Testcase 7	Easy	Hidden case	Success	10	0.1031 sec	8.82 KB
Testcase 8	Easy	Hidden case	Success	10	0.0426 sec	8.82 KB
Testcase 9	Easy	Hidden case	Success	10	0.1512 sec	8.96 KB

No Comments

PDF generated at: 20 Sep 2023 09:36:01 UTC