**PROBLEM SOLVING**

(Solving Various Problems Using C Language)

*Summer Internship Report Submitted in partial fulfillment*

*of the requirement for under graduate degree of*

**Bachelor of Technology**

In

**Computer Science And Engineering**

By

**Syed Irfan**

**221710313055**

[**https://github.com/SYEDIRFAN555/PROBLEM-SOLVING-221710313055**](https://github.com/SYEDIRFAN555/PROBLEM-SOLVING-221710313055)

*Under the Guidance of*

Assistant Professor



Department Of Computer Science And Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

                                                        July 2020

**DECLARATION**

I submit this industrial training work entitled **“SOLVING VARIOUS PROBLEMS USING C LANGUAGE**” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “Computer Science And Engineering”. I declare that it was carried out independently by me under the guidance of  **Mr.** , Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD                 SYED IRFAN

Date:                                                                                                       221710313055

GITAM (DEEMED TO BE UNIVERSITY)

     Hyderabad-502329, India

                                                                 Dated: 

**CERTIFICATE**

 This is to certify that the Industrial Training Report entitled "**SOLVING VARIOUS PROBLEMS USING C LANGUAGE"** is being submitted by SYED IRFAN (221710313055) in partial fulfillment of the requirement for the award of **Bachelor of Technology** **in Computer Science & Technology Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-20

It is faithful record work carried out by him at the **Computer Science And Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

**Mr.                                     Dr.Phani kumar**

Assistant Professor                                                                 Professor  and HOD

Department of CSE                                                            Department of CSE

**ACKNOWLEDGEMENT**

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank Dr. **N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Dr.SITARAMAYYA,** Principal, GITAM Hyderabad

I would like to thank respected **Dr.Phani Kumar,** Head of the Department of Computer Science And Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties **Mr.** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

                                                                             SYED IRFAN

221710313055

**TABLE OF CONTENTS**

**1 Introduction to the project 7**

**2  Problem 1 8**

2.1 Problem Statement:- 8

2.2 Coding 9

2.3 Output 11

**3  Problem 2 12**

3.1 Problem Statement:- 12

3.2 Coding 13

3.3 Output 15

**4  Problem 3 16**

4.1 Problem Statement:- 16

4.2 Coding 17

4.3 Output 19

**5  Problem 4 20**

5.1 Problem Statement:- 20

5.2 Coding 21

5.3 Output 25

**6  Problem 5 26**

6.1 Problem Statement:- 26

6.2 Coding 27

6.3 Output 30

**7  Problem 6 31**

7.1 Problem Statement:- 31

7.2 Coding 32

[7.3 Output](https://docs.google.com/document/d/1tjcjW_gBxLXJJ8lcM_zoOmIZgnojR5jVKL0I-zXZKt8/edit#heading=h.2bn6wsx)  35

**8  Software Requirements 36**

8.1  Hardware Requirements 36

8.2 Software Requirements 36

**9 Bibliography 37**

**1 Introduction to the project**

Problem Solving is the Process of Designing and carrying out certain steps to reach a Solution. SIX problems which are listed below are of different complexity and require different approach and logics in order to achieve desired Output/ Solution

1.THE LOST BUNNY-In this problem we have to find home for the bunny by giving it sutaible

Directions till it reaches its home.

2.THE MONEY BANK-In this problem we calculate a person's account balance after an

attempted transaction

3.MEMORY TEST- In this problem we follow order to check the correctness, Captain asks to produce the correct values for each soldier (That is the correct value each soldier should shout out).

4.READ EITHER WAY-In this problem we have to find More precisely, the nearest Palindrome greater than the given number.

5.BUILD THE TRIANGLE-In this problem we have to form a triangle with wooden sticks of various lengths and should succeed in order to win.

6.THE TWO NUMBERS- In this problem the program should accept 10 non-negative integers as input and output the number of distinct non-negative integers in the input.

I have executed projects in C language and For C language, I have used DEV C++ to execute the codes .

2. PROBLEM 1

THE LOST BUNNY

2.1 PROBLEM STATEMENT:-

Robot Bunny is lost. It wants to reach its home as soon as possible. Currently it is standing at coordinates (x1, y1) in 2-D plane. Its home is at coordinates (x2, y2). Bunny is extremely worried. Please help it by giving a command by telling the direction in which it should go so as to reach its home. If you give it a direction, it will keep moving in that direction till it reaches its home. There are four possible directions you can give as command - "left", "right", "up", "down". It might be possible that you can't instruct the robot in such a way that it reaches its home. In that case, output "sad".

**Input:**

First line of the input contains an integer T denoting the number of test cases. T test cases follow.

First line of each test case contains four space separated integers x1, y1, x2, y2.

**Output:**

For each test case, output a single line containing "left" or "right" or "up" or "down" or "sad" (without quotes).

**Constraints:**

1 ≤ T ≤ 5000

0 ≤ x1, y1, x2, y2. ≤ 100

It's guaranteed that the initial position of robot is not his home.

**Example:**

**Input:**

3

0 0 1 0

0 0 0 1

0 0 1 1

**Output:**

right

up

sad

**Explanation:**

**Test case 1.** If you give Bunny the command to move to the right, it will reach its home.

Concepts Used To Solve:-

Void Functions-Void functions are created and used just like value-returning functions except they do not return a value after the function executes. In lieu of a data type, void functions use the keyword "void." A void function performs a task, and then control returns back to the caller--but, it does not return a value.

Return Value:- Void functions are “void” due to the fact that they are not supposed to return values. True, but not completely. We cannot return values but there is something we can surely return from void functions. Some of cases are listed below.

2.2 CODING

#include <stdio.h>

void RobotDirection(int x1,int y1,int x2,int y2 ){

if(x1==x2)

{

if(y1>y2)

printf("down\n");

else

printf("up\n");

}

else if(y1==y2)

{

if(x1>x2)

printf("left\n");

else

printf("right\n");

}

else

printf("sad\n");

}

int main()

{

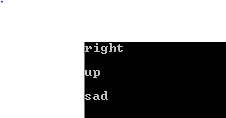
RobotDirection(0,0,1,0);

printf("\n");

RobotDirection(0,0,0,1);

printf("\n");

RobotDirection(0,0,1,1);

2.3 OUTPUT

**Fig 2.1.1**

3.PROBLEM 2

THE MONEY BANK

3.1 PROBLEM STATEMENT:-

Pooja would like to withdraw X $US from an ATM. The cash machine will only accept the transaction if X is a multiple of 5, and Pooja's account balance has enough cash to perform the withdrawal transaction (including bank charges). For each successful withdrawal the bank charges 0.50 $US. Calculate Pooja's account balance after an attempted transaction.

**Input:**

Positive integer 0 < X <= 2000 - the amount of cash which Pooja wishes to withdraw.

Nonnegative number 0<= Y <= 2000 with two digits of precision - Pooja's initial account balance.

**Output:**

Output the account balance after the attempted transaction, given as a number with two digits of precision. If there is not enough money in the account to complete the transaction, output the current bank balance.

**Example - Successful Transaction**

**Input:**

30 120.00

**Output:**

89.50

**Example - Incorrect Withdrawal Amount (not multiple of 5)**

**Input:**

42 120.00

**Output:**

120.00

**Example - Insufficient Funds**

**Input:**

300 120.00

**Output:**

120.00

Concepts Used To Solve:-

Void Functions-Void functions are created and used just like value-returning functions except they do not return a value after the function executes. In lieu of a data type, void functions use the keyword "void." A void function performs a task, and then control returns back to the caller--but, it does not return a value.

Return Value:- Void functions are “void” due to the fact that they are not supposed to return values. True, but not completely. We cannot return values but there is something we can surely return from void functions. Some of cases are listed below.

Float-Float is a data type which is used to represent the floating point numbers. It is a 32-bit IEEE 754 single precision floating point number ( 1-bit for the sign, 8-bit for exponent, 23\*-bit for the value. It has 6 decimal digits of precision.

3.2 CODING

#include <stdio.h>

#include <stdlib.h>

void BankTranaction(int wth,float ava){

float bal;

if((ava-wth)<0.5)

printf("%.2f",ava);

else

{

if(((wth%5)==0)) //if(((x%5)==0) && ((y-x)>0.5))

{

bal=(ava-wth-0.5);

printf("%.2f",bal);

}

else

printf("%.2f",(ava));

}

}

int main()

{

BankTranaction(30,120.00);

printf("\n");

BankTranaction(42,120.00);

printf("\n");

BankTranaction(300,120.00);

return 0;

}

3.3 OUTPUT



**Fig 3.1.1**

4.PROBLEM 3

MEMORY TEST

4.1 PROBLEM STATEMENT:-

N Soldiers are lined up for a memory test. They are numbered from 0 to N-1 from left to right.

In the test, there are M rounds. In each round, the Captain selects one position. Soldiers at that position will be numbered 0. All the soldiers to the right of the selected position will be numbered one greater than the soldier to his left. All the soldiers to the left of the selected position will be numbered one greater than the soldier to his right.

eg. if N = 6 and the selected position is 3, then the numbering will be [3, 2, 1, 0, 1, 2].

After M rounds, Captain asked each soldier to shout out the greatest number he was assigned during the M rounds. In order to check the correctness, Captain asked you to produce the correct values for each soldier (That is the correct value each soldier should shout out).

**Input:**

The first line of the input contains an integer T denoting the number of test cases.

First line of each test case contains two integers, N and M.

Second line of each test case contains M integers, the positions selected by Captain, in that order.

**Output:**

For each test case, output one line with N space separated integers.

**Constraints:**

1 ≤ T ≤ 10^4

1 ≤ N ≤ 10^5

1 ≤ M ≤ 10^5

1 ≤ Sum of N over all test cases ≤ 10^5

1 ≤ Sum of M over all test cases ≤ 10^5

0 ≤ Positions selected by captain ≤ N-1

**Example:**

**Input:**

2

4 1

1

6 2

2 3

**Output:**

1 0 1 2

3 2 1 1 2 3

Concepts Used To Solve:-

Max And Min Values:- Declare two variables max and min to store maximum and minimum. Assume first array element as maximum and minimum both, say max = arr[0] and min = arr[0] . Iterate through array to find maximum and minimum element in array. Run loop from first to last array element i.e. 0 to size - 1

4.2 CODING

#include<stdio.h>

#include<math.h>

int mx(int a,int b)

{

return ((a>b)?a:b);

}

int main()

{

int t;

scanf("%d",&t);

while(t--)

{

int n,m,i;

scanf("%d%d",&n,&m);

int a[m],j;

int max=0,min=n-1;

for(i=0;i<m;i++)

{

scanf("%d",&a[i]);

if(a[i]>max) max=a[i];

if(a[i]<min) min=a[i];

}

for(i=0;i<n;i++)

printf("%d ",mx(abs(i-max),abs(i-min)));

printf("\n");

}

return 0;

}

4.3 OUTPUT

**Fig 4.1.1**

5.PROBLEM 4

READ EITHER WAY

5.1 PROBLEM STATEMENT:-

One day, Riya challenged Ratan to find the numbers which are read the same from left to right or from right to left. She tells him that such numbers are known as Palindromes, and gives him a number and asks the smallest palindrome which is larger than the number. More precisely, the nearest Palindrome is greater than the given number. Help Ratan in figuring out the way to solve the challenge.

**Note**: Numbers are always displayed without leading zeros.

**Input:**

A number(K)

**Output:**

For each K, output the smallest palindrome larger than K.

**Example:**

**Input: Output:**

808 818

2133 2222

Concepts Used To Solve:-

Flag- a flag is a value that acts as a signal for a function or process. The value of the flag is used to determine the next step of a program. Flags are often binary flags, which contain a boolean value (true or false). However, not all flags are binary, meaning they can store a range of values. ... flag).

strlen- strlen( ) function counts the number of characters in a given string and returns the integer value. It stops counting the character when a null character is found. Because, null character indicates the end of the string in **C**.

Return- The strlen() function takes a string as an argument and returns its length. The returned value is of type size\_t (the unsigned integer type). It is defined in the <string. h> header file.

Char- char is the most basic data type in c. It stores a single character and requires a single byte of memory in almost all compilers.

**Usercode:** void Nearestpalindrome( char a[])

5.2 CODING

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void Nearestpalindrome( char a[])

{

int i,flag,check,l,m,k;

flag=1;

l=strlen(a);

k=l>>1;

check=0;

for(i=0;i<l;i++)

{

if(a[i]!='9')

{

flag=0;

break;

}

}

if(flag==1)

{

a[0]='1';

for(i=1;i<l;i++)

a[i]='0';

a[l]='1';

a[l+1]='\0';

check=1;

}

flag=0;

if(check!=1)

{

for(i=0;i<k;i++)

{

if(a[i]<a[l-1-i])

flag=-1;

else if(a[i]>a[l-1-i])

flag=1;

a[l-1-i]=a[i];

}

if(!(l&1))

m=k-1;

else m=k;

if(flag==0||flag==-1)

{

i=0;

while(a[m-i]=='9')

{

a[m-i]='0';

a[l-1-m+i]='0';

i++;

}

a[m-i]++;

a[l-1-m+i]=a[m-i];

}

}

printf("%s",a);

}

int main()

{

char a[100]="909";

Nearestpalindrome(a);

printf("\n");

char b[]="999";

Nearestpalindrome(b);

printf("\n");

char c[]="2133";

Nearestpalindrome(c);

printf("\n"); char d[]="1000";

Nearestpalindrome(d);

printf("\n");

char e[]="2432";

Nearestpalindrome(e);

printf("\n");

char a1[]="2569";

Nearestpalindrome(a1);

printf("\n");

char b1[]="1999";

Nearestpalindrome(b1);

printf("\n");

char c1[]="1889";

Nearestpalindrome(c1);

printf("\n");

return 0;

}

5.3 OUTPUT **Fig 5.1.1**

6 PROBLEM 5

BUILD THE TRIANGLE

6.1 PROBLEM STATEMENT:-

Neha has given Ritu, N (3 ≤ N ≤ 2,000) wooden sticks (which are labeled from 1 to N) and asked to pick three sticks at random. The i-th stick has a length of Li (1 ≤ Li ≤ 1,000,000). Neha has challenged Ritu, that if she is able to construct a triangle with those sticks (degenerate triangles included), she wins; otherwise, Ritu wins. Determine Ritu’s chances of winning by computing the number of ways she could choose three sticks (regardless of order) such that it is impossible to form a triangle with them.

**Input:**

Each test case starts with the single integer N, followed by a line with the integers L1, ..., LN. **Output:**

For each test case, output a single line containing the number of triples.

**Example:**

**Input:**

Number of sticks and their lengths.

**Output:**

Number of chances of winning, if degenerate , 0 else number of successful instances

**Explanation:**

For the first test case, 4 + 2 < 10, so you will win with the one available triple. For the second case, 1 + 2 is equal to 3; since degenerate triangles are allowed, the answer is 0.

Concept Used To Solve:-

Quicksort- QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways. 19

1. Always pick the first element as pivot.
2. Always pick last element as pivot (implemented below)
3. Pick a random element as pivot.
4. Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

6.2 CODING

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void quicksort(int \*arr,int start ,int end)

{

int i, j,key;

key = arr[(start + end)/2];

i = start;

j = end;

do

{

while( arr[i] < key )

i++;

while(arr[j] > key)

j--;

if(i<=j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

}while(i<=j);

if(start < j)

quicksort(arr,start,j);

if(i<end)

quicksort(arr,i,end);

}

void chances\_winning( int L[],int N)

{

int W,i,j,k;

quicksort(L,1,N);

W=0;

for(i=N;i>=3;i--)

{

j=1;

k=i-1;

while(j<k)

if(L[j]+L[k]<L[i])

{

W+=k-j;

j++;

}

else

k--;

}

printf("%d",W);

}

int main()

{

int A[]={1,2,3};

int M[]={2,3,10};

int N[]={1,2,1};

int O[]={5,6,2,7};

int B[]={1,5,6,9,10,15};

int C[]={1,2,3,4,5,6,7,8,9,10};

int D[]={15,19,29,39,48,49,59,58,68,69,25};

chances\_winning(A,3);

printf("\n");

chances\_winning(M,3);

printf("\n");

chances\_winning(N,3);

printf("\n");

chances\_winning(O,4);

printf("\n");

chances\_winning(B,6);

printf("\n");

chances\_winning(C,10);

printf("\n");

chances\_winning(D,11);

printf("\n");

return 0;

}

6.3 OUTPUT



**Fig 6.1.1**

7 .PROBLEM 6

THE TWO NUMBERS

7.1 PROBLEM STATEMENT:-

Given two integers 'A' and 'B', A MOD B is the remainder of the operation A / B.

For example,

7 MOD 3 = 1

14 MOD 3 = 2

27 MOD 3 = 0

38 MOD 3 = 2

Your program should accept 10 non-negative integers as input and output the number of distinct non-negative integers in the input, if they are considered MOD 42.

**Input:**

The input has 10 test cases. Each Test Case contains 10 non-negative integers, each smaller than 1000, one per line.

**Output:**

Output the number of distinct values when considered MOD 42, one per line for each test case.

**Example:**

**Input:**

39

40

41

42

43

44

82

83

84

85

... 9 more cases

**Output:**

6

... 9 more answers

Concept Used To Solve:-

Count- count() returns number of occurrences of an element in a given range. Returns the number of elements in the range [first,last) that compare equal to val. first, last : Input iterators to the initial and final positions of the sequence of elements.

Temp- A template is a simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don’t need to write the same code for different data types. For example, a software company may need sort() for different data types. Rather than writing and maintaining the multiple codes, we can write one sort() and pass data type as a parameter.

7.2 CODING

#include<stdio.h>

/\* Consider Size value always 10 \*/

void ModofTwoNumbers(int a[],int Size)

{

int i,count[10],c,j,temp;

c=0;

for(i=0;i<Size;i++){

a[i]=a[i]%42;

}

for(i=0;i<Size;i++){

count[i]=1;

}

for(i=0;i<Size;i++)

{

for(j=0;j<Size-1;j++)

{

if(a[j]>=a[j+1])

{

temp=a[j+1];

a[j+1]=a[j];

a[j]=temp;

}

}

}

for(i=0;i<Size;i++)

{

for(j=i+1;j<Size;j++)

{

if(a[i]==a[j])

{

count[i]--;

break;

}

}

}

for(i=0;i<Size;i++)

{

if(count[i]==1)

c++;

}

printf("%d\n",c);

}

int main()

{

int a[10]={39,40,41,42,43,44,82,83,84,85};

ModofTwoNumbers(a,10);

printf("\n");

int b[10]={29,58,59,60,72,73,75,83,84,85};

ModofTwoNumbers(b,10);

printf("\n");

int c[10]={225,226,227,335,251,337,447,448,489,555};

ModofTwoNumbers(c,10);

printf("\n");

int c1[10]={191,20,21,292,65,66,67,99,98,97};

ModofTwoNumbers(c1,10);

printf("\n");

int c2[10]={115,116,117,120,130,140,150,160,170,180};

ModofTwoNumbers(c2,10);

printf("\n");

return 0;

}

7.3 OUTPUT



**Fig 7.1.1**

**8  Software Requirements**

## 8.1  Hardware Requirements

This project can be executed in any system or an android phone without prior to any platform.

We can use any online compiler and interpreter.

## 8.2 Software Requirements

There are two ways to execute this projects

1. Online compilers
2. Softwares for execution (DEV C++.)

Online Compilers require only internet connection. We have many free compilers with which we can code.

Softwares for execution need to be installed based on the user’s system specification. These help us to completely execute the project. These softwares are based on the platforms

**9 BIBLIOGRAPHY**

https://drive.google.com/drive/u/0/mobile/folders/1JU1gRmlxffnToOxETYDBvOVXXPa6vt1R?usp=sharing