

# **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

**Vaddi Nymisha**

**221710307060**

*Under the Guidance of*



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, 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

VADDI NYMISHA

Date:

221710307060



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 VADDI NYMISHA (221710307060) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-20

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

**Dr. S. Phani Kumar**

Professor and HOD

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 respected **Dr. N. Siva Prasad**, Pro Vice Chancellor, GITAM Hyderabad and, Principal **Dr. N. Seetharamaiah**, GITAM Hyderabad

I would like to thank respected **Dr. S. 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 of what we study for.

I would like to thank the respected faculties 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.

VADDI NYMISHA

221710307060

# TABLE OF CONTENTS

<b>1 Introduction to the project</b>	<b>1</b>
<b>2 Problem 1 – Philaland Coins Problem</b>	
2.1 Problem Statement	2
2.2 Coding	4
2.3 Output	5
<b>3 Problem 2 – Catch-22</b>	
3.1 Problem Statement	6
3.2 Coding	9
3.3 Output	11
<b>4 Problem 3 – Super ASCII String Checker</b>	
4.1 Problem Statement	12
4.2 Coding	14
4.3 Output	15
<b>5 Problem 4 – Reverse Gear</b>	
5.1 Problem Statement	16
5.2 Coding	18
5.3 Output	19
<b>6 Problem 5 – The Vita Sum</b>	
6.1 Problem Statement	20
6.2 Coding	22
6.3 Output	23
<b>7 Problem 6 – Saving for a Rainy day</b>	
7.1 Problem Statement	24
7.2 Coding	26
7.3 Output	27
<b>8 Software Requirements</b>	
8.1 Hardware Requirements	28
8.2 Software Requirements	28
<b>BIBLIOGRAPHY</b>	<b>29</b>

# 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. **Philaland Coins problem** – In this problem we find the minimum number of denominations for any arbitrary maximum price.
2. **Catch-22** – In this problem we calculate the amount of time taken, before the Robot falls in a ditch, if at all it falls.
3. **Super ASCII String Checker** – In this problem we check whether the given string is a super ASCII string i.e., if the count of each character in the string is equal to its ASCII value, or not.
4. **Reverse Gear** – In this problem we find the amount of time available before an automated car hits the wall.
5. **The Vita Sum** – In this problem we find the sum of all the combinations of balls that can be pulled out of the bag when the number of balls picked is even.
6. **Saving for a Rainy day** – In this problem we find the required amount to be deposited for the assured cash flow for a fixed period in future at a given rate of interest at which the money will increase during the time period.

I have executed projects in C language. For executing the codes I have used CODE BLOCKS.

## 2 Problem 1

### Philaland Coins Problem

#### 2.1 Statement:

The problem solvers have found a new Island for coding and named it as Philaland. These smart people were given a task to make purchase of items at the Island easier by distributing various coins with different value. Manish has come up with a solution that if we make coins category starting from \$1 till the maximum price of item present on Island, then we can purchase any item easily. He added following example to prove his point.

Let's suppose the maximum price of an item is 5\$ then we can make coins of {\$1, \$2, \$3, \$4, \$5} to purchase any item ranging from \$1 till \$5.

Now Manisha, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {\$1, \$2, \$3}.

According to him any item can be purchased one time ranging from \$1 to \$5. Everyone was impressed with both of them. Your task is to help Manisha come up with minimum number of denominations for any arbitrary max price in Philaland.

#### Input Format

First line contains an integer T denoting the number of test cases. Next T lines contain an integer N denoting the maximum price of the item present at Philaland.

#### Output Format

For each test case print a single line denoting the minimum number of denominations of coins required.

#### Constraints

$$1 \leq T \leq 100$$

$$1 \leq N \leq 5000$$

Refer the Sample Output Formatting

#### Sample Input:

2



10

5

**Sample Output:**

4

3

**Explanation:**

For test case 1, N=10.

According to Manish {\$1, \$2, \$3,... \$10} must be distributed.

But as per Manisha only {\$1, \$2, \$3, \$4} coins are enough to purchase any item ranging from \$1 to \$10. Hence minimum is 4. Likewise denominations could also be {\$1, \$2, \$3, \$5}. Hence answer is still 4.

For test case 2, N=5.

According to Manish {\$1, \$2, \$3, \$4, \$5} must be distributed.

But as per Manisha only {\$1, \$2, \$3} coins are enough to purchase any item ranging from \$1 to \$5. Hence minimum is 3. Likewise denominations could also be {\$1, \$2, \$4}. Hence answer is still 3.

**Concepts Used:**

**For-loop:** A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly. It is used when the number of iterations is known.

**Syntax:**

```
for(init;condition;increment/decrement)
{
    Statements;
}
```

Here the for-loop is used for iterating the process for T test cases.

**While loop:** A while loop is a control flow statement that allows the code to be executed repeatedly based on a given Boolean condition.

**Syntax:**

```
while(condition)
{
    Statements;
}
```


Here the while loop is used to repeat the process till the maximum price N is greater than 0.

## 2.2 Coding

```
#include<stdio.h>
void main()
{
    int T,N,i,remainder,count=0,binary=0,b=1;
    // T = Number of Test Cases
    scanf("%d",&T);
    for(i=0;i<T;i++)
    {
        // N = The Maximum Price
        scanf("%d",&N);
        count=0;
        while(N>0)
        {
            remainder=N%2;
            // Number of digits in the binary number
            count++;
            // Binary number
            binary=binary+remainder*b;
            N=N/2;
            b=b*10;
        }
        printf("%d\n",count);
    }
    return 0;
}
```

Fig 2.2.1

## 2.3 Output



2  
10  
4  
5  
3

**Fig 2.3.1**

## 3 Problem 2

### Catch-22

#### 3.1 Statement

Catch-22, A robot is programmed to move forward  $F$  meters and backwards again, say  $B$  meters, in a straight line. The Robot covers 1 meter in  $T$  units of time. On Robot's path there is a ditch at a distance  $FD$  from initial position in forward direction as well as a ditch at a distance  $BD$  from initial position in backward direction. This forward and backward movement is performed repeatedly by the Robot. Your task is to calculate amount of time taken, before the Robot falls in either ditch, if at all it falls in a ditch.

#### Input Format:

First line contains total number of test cases, denoted by  $N$  Next  $N$  lines, contain a tuple containing 5 values delimited by space  $F B T FD BD$ , where

$F$  denotes forward displacement in meters

$B$  denotes backward displacement in meters

$T$  denotes time taken to cover 1 meter

$FD$  denotes distance from Robot's starting position and the ditch in forward direction

$BD$  denotes distance from Robot's starting position and the ditch in backward direction

#### Output Format:

For each test case print time taken by the Robot to fall in the ditch and also state which ditch he falls into. Print  $F$  for forward and  $B$  for backward. Both the outputs must be delimited by whitespace OR Print No Ditch if the Robot does not fall in either ditch

#### Constraints:

First move will always be in forward direction  $1 \leq N \leq 100$  forward displacement  $> 0$  backward displacement  $> 0$  time  $> 0$  distance of ditch in forward direction ( $FD$ )  $> 0$  distance of ditch in backward direction ( $BD$ )  $> 0$  All input values must be positive integers only  
Sample Input and Output

#### Sample Input and Output:

1.

3

9 4 3 13 10

9 7 1 11 13

4 4 3 8 12

63F 25F No Ditch

2.

5

8 4 7 11 22

4 5 4 25 6

4 9 3 6 29

7 10 6 24 12

10 10 1 9 7

133F 216B 231B 408B 9F

### **Concepts Used:**

**For-loop:** A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly. It is used when the number of iterations is known.

#### **Syntax:**

```
for(init;condition;increment/decrement)
{
    Statements;
}
```

Here the for-loop is used for iterating the process for N test cases.

**If-else-if ladder:** The if-else-if statement is used to execute a code with multiple conditions. It is a chain of if-else statements in which each if statement is associated with else if statement and last would be an else statement.

**Syntax:**

```
if(condition 1)
{
    Statements;
}
else if(condition 2)
{
    Statements;
}
.
.
else(condition n)
{
    Statements;
}
```

Here the if-else-if ladder is used to check the positions of the ditch and the movement of the robot.

**While loop:** A while loop is a control flow statement that allows the code to be executed repeatedly based on a given Boolean condition.

**Syntax:**

```
while(condition)
{
    Statements;
}
```

Here the while loop is used to repeat the process.

## 3.2 Coding

```
#include<stdio.h>
void main()
{
    int N,F,B,T,FD,BD,pos,total,i;
    //N = Number of test cases
    scanf("%d",&N);
    for(i=0;i<N;i++)
    {
        //total = Total number of meters
        total=0;
        // pos = present position
        pos=0;
        scanf("%d%d%d%d%d",&F,&B,&T,&FD,&BD);
        if(F>=FD)
        {
            pos=FD*T;
            printf("%dF\n",pos);
        }
        else if(F==B)
        {
            printf("No Ditch\n");
        }
        else if(F>B)
        {
            while(1)
            {
```

```

        pos=pos+F;
        total=total+F;
        if(pos>=FD)
            break;
        pos=pos-B;
        total=total+B;
    }
    total=total-(pos-FD);
    printf("%dF\n",total*T);
}
else
{
    while(1)
    {
        pos=pos-F;
        total=total+F;
        pos=pos+B;
        total=total+B;
        if(pos>=BD)
            break;
    }
    total=total-(pos-BD);
    printf("%dB\n",total*T);
}
}
return 0;
}

```

**Fig 3.2.1**



### 3.3 Output

```
3
9 4 3 13 10
63F
9 7 1 11 13
25F
4 4 3 8 12
No Ditch
```

Fig 3.3.1

```
5
8 4 7 11 22
133F
4 5 4 25 6
216B
4 9 3 6 29
231B
7 10 6 24 12
408B
10 10 1 9 7
9F
```

Fig 3.3.2

## 4 Problem 3

### 1. Super ASCII String Checker

#### 4.1 Statement

In the Byteland country a string "S" is said to super ASCII string if and only if count of each character in the string is equal to its ASCII value. In the Byteland country ASCII code of 'a' is 1, 'b' is 2 ... 'z' is 26. Your task is to find out whether the given string is a super ASCII string or not.

#### Input Format:

First line contains number of test cases T, followed by T lines, each containing a string "S".

#### Output Format:

For each test case print "Yes" if the String "S" is super ASCII, else print "No"

#### Constraints:

$1 \leq T \leq 100$

$1 \leq |S| \leq 400$ , S will contains only lower case alphabets ('a'-'z').

#### Sample Input and Output

2

bba

scca

YES

NO

#### Explanation:

In case 1, viz. String "bba" -

The count of character 'b' is 2. ASCII value of 'b' is also 2.

The count of character 'a' is 1. ASCII value of 'a' is also 1.

Hence string "bba" is super ASCII.

**Concepts used:**

**For-loop:** A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly. It is used when the number of iterations is known.

**Syntax:**

```
for(init;condition;increment/decrement)
{
    Statements;
}
```

Here the for-loop is used for iterating the process for T test cases and for the array a[].

**If-else statement:** The if-else statement is used to execute the code if condition is true or false.

**Syntax:**

```
if(expression)
{
    Statements
}
else
{
    Statements
}
```

Here the if-else statements are used to check if the number of alphabets is equal to its ASCII value or not.

**ASCII values** - American Standard Code for Information Interchange: It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127

**Header – string.h:** It defines one variable type, one macro and various functions for manipulating arrays of characters.

String function used: String length (strlen);

String length (strlen), computes the length of the string.

```
strlen(str);
```

**Arrays:** An array is a collection of data items of the same type.

Here array `a[]` is used to store the count of the repeated alphabets.

## 4.2 Coding

```
#include<stdio.h>
#include<string.h>
void main()
{
    int T,a[26],i,j,n,flag;
    char str[400];
    // T = Number of test cases
    scanf("%d",&T);
    for(i=0;i<T;i++)
    {
        for(j=0;j<26;j++)
        {
            a[j]=0;
        }
        scanf("%s",str);
        for(j=0;j<strlen(str);j++)
        {
            n=str[j]-97;
            a[n]++;
        }
        for(j=0;j<26;j++)
        {
            if(a[j]==j+1||a[j]==0)
            {
                flag=1;
            }
            else
                .
        }
    }
}
```

```

        {
            flag=0;
            break;
        }
    }
    if(flag==1)
    {
        printf("YES\n");
    }
    else
    {
        printf("NO\n");
    }
}
return 0;
}

```

Fig 4.2.1

### 4.3 Output

```

2
bba
YES
scca
NO

```

Fig 4.3.1

## 5 Problem 4

### Reverse Gear

#### 5.1 Statement:

A futuristic company is building an autonomous car. The scientists at the company are training the car to perform Reverse parking. To park, the car needs to be able to move in backward as well as forward direction. The car is programmed to move backwards B meters and forwards again, say F meters, in a straight line. The car does this repeatedly until it is able to park or collides with other objects. The car covers 1 meter in T units of time. There is a wall after distance D from car's initial position in the backward direction.

The car is currently not without defects and hence often hits the wall. The scientists are devising a strategy to prevent this from happening. Your task is to help the scientists by providing them with exact information on amount of time available before the car hits the wall.

#### Input Format:

First line contains total number of test cases, denoted by N

Next N lines, contain a tuple containing 4 values delimited by space

F B T D, where

F denotes forward displacement in meters

B denotes backward displacement in meters

T denotes time taken to cover 1 meter

D denotes distance from Car's starting position and the wall in backward direction

#### Output Format:

For each test case print time taken by the Car to hit the wall

Constraints:

First move will always be in backward direction

$1 \leq N \leq 100$

backward displacement > forward displacement i.e.  $(B > F)$

forward displacement (F) > 0

backward displacement (B) > 0

time (T) > 0

distance (D) > 0

All input values must be positive integers only

### **Sample Input and Output**

6 9 3 18

3 7 5 20

162

220

### **Concepts used:**

**For-loop:** A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly. It is used when the number of iterations is known.

#### **Syntax:**

```
for(init;condition;increment/decrement)
{
    Statements;
}
```

Here the for-loop is used for iterating the process for N test cases and for iterating the process to determine the direction of the movement.

**If-else statement:** The if-else statement is used to execute the code if condition is true or false.

#### **Syntax:**

```
if(expression)
{
    Statements
}
else
{
    Statements
}
```

Here the if-else statements are used to determine the direction of the movement.

## 5.2 Coding

```
#include<stdio.h>
int main()
{
    int N,F,B,T,D,i,j,b=0,f=0,total;
    // N = Number of test cases
    scanf ("%d",&N);
    for(i=0;i<N;i++)
    {
        b=0;
        f=0;
        scanf ("%d%d%d%d",&F,&B,&T,&D);
        for(j=0;D>0;j++)
        {
            //To determine the direction of the movement
            if(j%2==0)
            {
                D=D-B;
                b++;
            }
            else
            {
                D=D+F;
                f++;
            }
        }
        b=b*B;

        f=f*F;
        total=(b+f)*T+(D*T);
        printf("%d\n",total);
    }
    return 0;
}
```

Fig 5.2.1



### 5.3. Output

```
2
6 9 3 18
162
3 7 5 20
220
```

Fig 5.3.1

## 6 Problem 5

### The Vita Sum

#### 6.1 Statement:

Tom the cat is brushing up his Math skills. He has a bag containing  $N$  balls of different colors. Now Tom can randomly pick any even number of balls from the bag. Tom wants to find out the sum of all such combinations of balls that he can pull out from the bag. Given he can pull out at max  $K$  balls in one pick.

#### Input Format:

First line contains two space separated numbers  $N$  and  $K$

#### Output Format:

The output is the sum of all the combinations of balls he can pull out modulo  $10^9+7$  i.e. (1000000007)

#### Constraints:

$$0 \leq N, k \leq 10^9$$

$$N \geq k$$

#### Sample Input and Output:

4 4

8

We need  $4C_0 + 4C_2 + 4C_4 = 1 + 6 + 1 = 8$

8 3

29

We need  $8C_0 + 8C_2 = 1 + 28 = 29$

#### Concepts used:

**For-loop:** A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly. It is used when the number of iterations is known.

**Syntax:**

```
for(init;condition;increment/decrement)
{
    Statements;
}
```

Here the for-loop is used for finding factorial of the number and to iterate the process to select the even number picks.

**If-else statement:** The if-else statement is used to execute the code if condition is true or false.

**Syntax:**

```
if(expression)
{
    Statements
}
else
{
    Statements
}
```

Here the if-else statements are used to select the even number picks.

**Function:** A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.

**Syntax:**

```
return_type function_name(parameter list)
{
    Body of the function;
}
```

Here, the functions used are int factorial(int num) and int ncr(int a, int b).

## 6.2 Coding

```
#include<stdio.h>
int factorial(int num)
{
    int fact=1,i;
    for(i=1;i<=num;i++)
    {
        fact=fact*i;
    }
    return fact;
}
int ncr(int a,int b)
{
    int x;
    x=(factorial(a)/(factorial(a-b)*factorial(b)));
    return x;
}
int main()
{
    int N,K,i,x=0;
    scanf("%d%d",&N,&K);
    for(i=1;i<=K;i++)
    {
        if(i%2==0)
        {
            x=x+ncr(N,i);
        }
        else
        {
            continue;
        }
    }

    printf("%d",x+1);
    return 0;
}
```

Fig 6.2.1

## 6.3 Output



```
4 4
8
```

Fig 6.3.1



```
8 3
29
```

Fig 6.3.2

## 7 Problem 6

### Saving for a Rainy day

#### 7.1 Statement:

By nature, an average Indian believes in saving money. Some reports suggest that an average Indian manages to save approximately 30+% of his salary. Dhaniram is one such hard working fellow. With a view of future expenses, Dhaniram resolves to save a certain amount in order to meet his cash flow demands in the future. Consider the following example. Dhaniram wants to buy a TV. He needs to pay Rs.2000/- per month for 12 installments to own the TV. If let's say he gets 4% interest per annum on his savings bank account, then Dhaniram will need to deposit a certain amount in the bank today, such that he is able to withdraw Rs.2000/- per month for the next 12 months without requiring any additional deposits throughout. Your task is to find out how much Dhaniram should deposit today so that he gets assured cash flows for a fixed period in the future, given the rate of interest at which his money will grow during this period.

#### Input Format:

First line contains desired cash flow M  
Second line contains period in months denoted by T  
Third line contains rate per annum R expressed in percentage at which deposited amount will grow

#### Output Format:

Print total amount of money to be deposited now rounded off to the nearest integer

#### Constraints:

$M > 0$   $T > 0$   $R \geq 0$  Calculation should be done upto 11-digit precision Sample Input and Output

500 3 12

1470

6000 3 5.9

17824

500 2 0

1000

**Concepts used:**

**While loop:** A while loop is a control flow statement that allows the code to be executed repeatedly based on a given Boolean condition.

**Syntax:**

```
while(condition)
{
    Statements;
}
```

Here, the while loop is used to repeat the process till months t is greater 0.

**If-else statement:** The if-else statement is used to execute the code if condition is true or false.

**Syntax:**

```
if(expression)
{
    Statements
}
else
{
    Statements
}
```

Here the if-else statements are used to find the round figure of the required amount.

**Header – math.h:** The math.h header defines various mathematical functions and one macro.

Math functions used:

**ceil(x):** Returns the smallest integer value greater than or equal to x.

```
ceil(required);
```

**floor(x):** Returns the largest integer value less than or equal to x.

```
floor(required);
```

## 7.2 Coding

```
#include<stdio.h>
#include<math.h>
int main()
{
    int M,t;
    float rate,amt,interest,required;
    scanf ("%d%d%f",&M,&t,&rate);
    required=M;
    while (t--)
    {
        amt=required/(1+rate/(float)1200);
        interest=required-amt;
        required+=M-interest;
    }
    required-=M;
    double finalAmount=ceil(required-0.5);
    if(finalAmount>required)
        finalAmount=ceil(required);
    else
        finalAmount=floor(required);
    printf("%.1f",finalAmount);
    return 0;
}
```

Fig 7.2.1



## 7.3 Output

```
500 3 12  
1470
```

Fig 7.3.1

```
6000 3 5.9  
17824
```

Fig 7.3.2

```
500 2 0  
1000
```

Fig 7.3.3

## **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 project

1. Online Compilers
2. Softwares for execution (CODE BLOCKS, DEV C++, ANACONDA...)

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 users system specification. These help us to completely execute the project. These softwares are based on the platforms.

## BIBLIOGRAPHY

1. Prateek Chauhan. “TCS CodeVita Philaland Coin” *programmersdoor.com*. Programmers Door, 20 Jun. 2020.
2. Vikash. “TCS CodeVita Season IV Round 1 Question: Catch 22” *programminggeek.in*. Programming Geek, 22 Jul. 2016.
3. Vikash. “TCS CodeVita 2014 Questions: Round 1 (Set1)” *programminggeek.in*. Programming Geek, 14 Sept. 2014.
4. Vikash. “CodeVita Season IV Round 1: Reverse Gear” *programminggeek.in*. Programming Geek, 8 Aug. 2015.
5. Vikash. “CodeVita 2016 Round 2 Question: The Vita Sum” *programminggeek.in*. Programming Geek, 20 Aug. 2016.
6. Vikash. “TCS CodeVita Season IV Round 1 Question: Saving for a rainy day” *programminggeek.in*. Programming Geek, 22 Jul. 2016.