

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

SIROBHUSHANAM BHAVYA

221710305054

<https://github.com/Bhavyasiro1>

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 **Mr.**, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

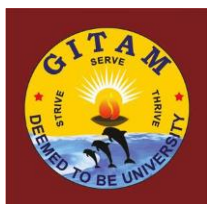
The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD

S.BHAVYA

Date:

221710305054



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 SIROBHUSHANAM BHAVYA (221710305054) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science & Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-2020.

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

Mr.

Assistant Professor

Department of CSE

Dr. 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 **Dr. CH. Sanjay**, Principal, GITAM Hyderabad

I would like to thank respected **Dr.**, 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 a internship report. It helped me a lot to realize of 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.

S.BHAVYA

221710305054

TABLE OF CONTENTS

1 Introduction to the project	1
2 Problem 1- Maneuvering a cave problem	2
2.1 Problem Statement	2
2.2 Coding	3
2.3 Output	4
3 Problem 2- The jumping kangaroo’s problem	5
3.1 Problem Statement	5
3.2 Coding	7
3.3 Output	8
4 Problem 3- The reverse game problem	9
4.1 Problem Statement	9
4.2 Coding	11
4.3 Output	12
5 Problem 4- Egg dropping puzzle	13
5.1 Problem Statement	13
5.2 Coding	15
5.3 Output	15
6 Problem 5- Lucky palindrome	16
6.1 Problem Statement	16
6.2 Coding	20
6.3 Output	20
7 Software Requirements	21
7.1 Hardware Requirements	
7.2 Software Requirements	
8 Bibliography	22

1. Introduction to the project

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

1. **Maneuvering a cave problem** - The problem is to count all the possible paths from top left to bottom right of a $m \times n$ matrix with constraints that from each cell you can either move only to right or down.
2. **The jumping kangaroo's problem** -In this problem the starting locations and movement rates are given for each kangaroo based on those we must be able to determine if they will ever land at the same location at the same time.
3. **The reverse game problem** - In this problem a person reverses the position of balls and the person should be able to determine the final position of any ball asked by other person.
4. **Egg dropping puzzle** – The problem is to decide the floors from which eggs should drop so that the total number of trials are minimized.
5. **Lucky palindrome** – The problem is to find the lexicographically smallest palindrome that can be constructed in a greedy manner using minimum number of replacement operations.

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

2. Problem- 1

MANEUVERING A CAVE PROBLEM

The problem is to count possible paths in a $m \times n$ matrix.

2.1 PROBLEM STATEMENT:

The problem is to count all the possible paths from top left to bottom right of a $m \times n$ matrix with the constraints that **from each cell you can either move only to right or down**. It is a Two-dimensional array manipulation problem.

INPUT:

First line consists of T test cases. First line of every test case consists of N and M, denoting the number of rows and number of columns respectively.

OUTPUT:

Single line output i.e count of all the possible paths from top left to bottom right of a $m \times n$ matrix.

Sample Input:

Enter the number of rows of the matrix: 3
Enter the number of columns of the matrix: 3

Sample Output:

6

CONCEPTS USED:

1. Functions
 2. If- else statement
- **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

}
```


- **If-else statement:** The if-else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given condition is true

Syntax:

```
If(test expression)

{

//statements

}

else

{

//statements

}
```

2.2 CODING:

```
cave.c
1  #include <stdio.h>
2  int calc(int x, int y) ;
3  int main()
4  {
5      int a,b;
6      printf("Enter the number of rows of the matrix : ");
7      scanf("%d",&a);
8      printf("Enter the number of columns of the matrix : ");
9      scanf("%d",&b);
10     printf("%d", calc(a,b));
11     return 0;
12 }
13 int calc(int x, int y)
14 {
15     if (x == 1 || y == 1)
16     {
17         return 1;
18     }
19     else
20     {
21         return calc(x - 1, y) + calc(x, y - 1);
22     }
23 }
```

Fig: 2.2.1

2.3 OUTPUT:

```
Enter the number of rows of the matrix : 3
Enter the number of columns of the matrix : 3
6
-----
Process exited after 6.501 seconds with return value 0
Press any key to continue . . .
```

Fig: 2.3.1

3. PROBLEM -2

THE JUMPING KANGAROOS PROBLEM

In this problem the starting locations and movement rates are given for each kangaroo based on those we must be able to determine if they will ever land at the same location at the same time.

3.1 PROBLEM STATEMENT:

There are two kangaroos on an x-axis ready to jump in the positive direction (i.e, toward positive infinity). The first kangaroo starts at location **X1** and moves at a rate of **V1** meters per jump. The second kangaroo starts at location **X2** and moves at a rate of **V2** meters per jump. Given the starting locations and movement rates for each kangaroo we should determine if they'll ever land at the same location at the same time.

The two kangaroos must land at the same location after making the same number of jumps.

INPUT:

A single line of four space-separated integers denoting the respective values of X1,V1, X2 and V2.

OUTPUT:

Print YES if they can land on the same location at the same time; otherwise, print NO.

Sample Input:

Enter the 1st location: 0

Enter the rate of movement for 1st location: 5

Enter 2nd location: 3

Enter the rate of movement for second location: 2

Sample Output:

YES

CONCEPTS USED:

1. If - else if – else statement

- **If-else if- else statement:**

The if-else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given conditions true.

Syntax:

```
if(Boolean expression)

{

/*executes when Boolean expression 1 is true*/

}

else if(Boolean expression 2)

{

/*executes when the boolean expression 2 is true*/

}

else if(Boolean expression 3)

{

/*executes when the boolean expression 3 is true*/

}

.

.

.

.

else

{

/*executes when the none of the above condition is true*/

}
```

3.2 CODING:

```
1  #include <stdio.h>
2  int main(void) {
3      int x1,x2,v1,v2,i;
4      printf("enter the 1st location : \n");
5      scanf("%d",&x1);
6      printf("enter the rate of movement for first location : \n");
7      scanf("%d",&v1);
8      printf("enter the 2nd location : \n");
9      scanf("%d",&x2);
10     printf("enter the rate of movement for 2nd location : \n");
11     scanf("%d",&v2);
12     if(v1>v2)
13     {
14         if(x1>x2)
15             printf("NO");
16         else{
17             if((x2-x1)%(v1-v2)==0)
18                 printf("YES");
19             else
20                 printf("NO");
21         }
22     }
23     else if (v2>v1){
24         if(x2>x1)
25             printf("NO");
26         else{
27             if((x1-x2)%(v2-v1)==0){
28                 printf("YES");
29             }
30             else{
31                 printf("NO");
32             }
33         }
34     }
35
36     return 0;
37 }
```

Fig: 3.2.1

3.3 OUTPUT:

```
enter the 1st location :
0
enter the rate of movement for first location :
3
enter the 2nd location :
4
enter the rate of movement for 2nd location :
2
YES
-----
Process exited after 14.33 seconds with return value
Press any key to continue . . .
```

Fig: 3.3.1

```
enter the 1st location :
0
enter the rate of movement for first location :
2
enter the 2nd location :
5
enter the rate of movement for 2nd location :
3
NO
-----
Process exited after 8.04 seconds with return value 0
Press any key to continue . . .
```

Fig: 3.3.2

4. PROBLEM – 3

THE REVERSE GAME PROBLEM

In this problem a person reverses the position of balls and the person should be able to determine the final position of any ball asked by other person.

4.1 PROBLEM STATEMENT:

Person A and Person B are playing a game. They have N balls numbered from 0 to N-1. Person A asks Person B to reverse the position of the balls, i.e., to change the order from say, 0,1,2,3 to 3,2,1,0. He further asks Person A to reverse the position of the balls N times, each time starting from one position further to the right, till he reaches the last ball. So, Person A has to reverse the positions of the ball starting from 0th position, then from 1st position, then from 2nd position and so on. At the end of the game, Person B will ask Person A the final position of any ball numbered K. Person A will win the game, if he can answer. Help the Person A.

This problem follows Brute force approach.

INPUT:

The first line contains an integer T, i.e., the number of the test cases.

The next T lines will contain two integers N and K.

OUTPUT:

Print the final index in array

Sample Input:

Enter the number of test cases: 2

Number of terms: 3 1

Number of terms: 5 2

Sample Output:

Index of a given numbers: 2

Index of a given numbers: 4

CONCEPTS USED:

1. For loop
2. If –else statement
3. break

- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for(initializationStatement ; testExpression ; updateStatement)

{

    // Statements

}
```

- **If-else if- else statement:** The if-else if- else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given conditions true

Syntax:

```
if(Boolean expression)

{

    /*executes when Boolean expression 1 is true*/

}

else if(Boolean expression 2)

{

    /*executes when the boolean expression 2 is true*/

}

.....

Else

{

    /*executes when the none of the above condition is true*/

}
```

- **Break:**

The break statement is a keyword in C which is used to bring the program control out of the loop the break statement is used inside loops or switch statement. The break statement breaks the loop one by one, i.e, in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break statement in c can be used in the following scenarios:

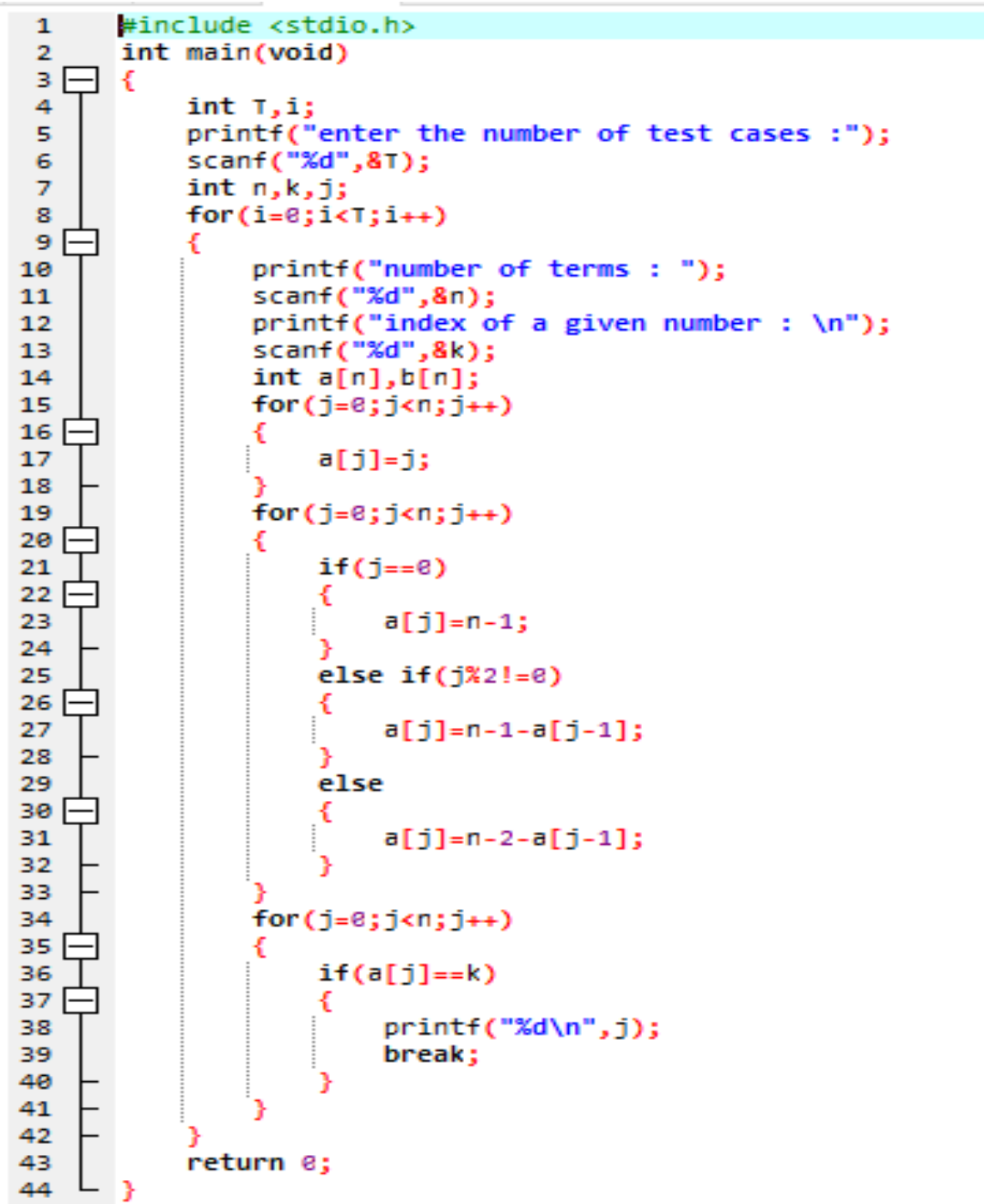
1. with switch case
2. with loop

Syntax:

//loop or switch case

Break;

4.2 CODING:



```

1  #include <stdio.h>
2  int main(void)
3  {
4      int T,i;
5      printf("enter the number of test cases :");
6      scanf("%d",&T);
7      int n,k,j;
8      for(i=0;i<T;i++)
9      {
10         printf("number of terms : ");
11         scanf("%d",&n);
12         printf("index of a given number : \n");
13         scanf("%d",&k);
14         int a[n],b[n];
15         for(j=0;j<n;j++)
16         {
17             a[j]=j;
18         }
19         for(j=0;j<n;j++)
20         {
21             if(j==0)
22             {
23                 a[j]=n-1;
24             }
25             else if(j%2!=0)
26             {
27                 a[j]=n-1-a[j-1];
28             }
29             else
30             {
31                 a[j]=n-2-a[j-1];
32             }
33         }
34         for(j=0;j<n;j++)
35         {
36             if(a[j]==k)
37             {
38                 printf("%d\n",j);
39                 break;
40             }
41         }
42     }
43     return 0;
44 }

```

Fig: 4.2.1

4.3 OUTPUT:

```
enter the number of test cases :2
number of terms : 3 1
index of a given number :
2
number of terms : 5 2
index of a given number :
4

-----
Process exited after 13.93 seconds with return value 0
Press any key to continue . . .
```

Fig: 4.3.1

5. PROBLEM- 4

EGG DROPPING PUZZLE

The problem is to decide the floors from which eggs should be dropped so that the total number of trials are minimized.

5.1 PROBLEM STATEMENT:

The problem is to decide the floors from which eggs should be dropped so that the total number of trials are minimized.

In this problem we will be finding a solution with 'n' eggs and 'k' floors. The solution is to try dropping an egg from every floor (from 1 to k) and recursively calculate the minimum number of droppings needed in the worst case. the floor which gives the minimum value in the worst case is going to be part of the solution.

INPUT:

K-> number of floors

n-> number of eggs

OUTPUT:

Minimum number of trials in worst case number of eggs and number of floors.

Sample Input :

Enter number of eggs and floors: 5 10

Sample Output:

Minimum number of trials in worst case
with 5 eggs and 10 floors is 4

CONCEPTS USED:

1. functions
2. if statement
3. for loop

- **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

}
```

- **If statement:**

The statements inside the body of “if” only execute if the given condition returns true. If the condition returns false then the statements inside “if” are skipped.

Syntax:

```
If(test expression)

{

//statements

}
```

- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for(initializationStatement ; testExpression ; updateStatement)

{

// Statements

}
```

5.2 CODING:

```
1  #include <limits.h>
2  #include <stdio.h>
3  int max(int a, int b)
4  {
5      return (a > b) ? a : b;
6  }
7
8  int eggDrop(int n, int k)
9  {
10
11      if (k == 1 || k == 0)
12          return k;
13
14      if (n == 1)
15          return k;
16
17      int min = INT_MAX, x, res;
18
19      for (x = 1; x <= k; x++) {
20          res = max(
21              eggDrop(n - 1, x - 1),
22              eggDrop(n, k - x));
23          if (res < min)
24              min = res;
25      }
26
27      return min + 1;
28  }
29
30  int main()
31  {
32      int n, k;
33      printf("enter number of eggs and floors : ");
34      scanf("%d %d",&n,&k);
35      printf("Minimum number of trials in "
36             "worst case with %d eggs and "
37             "%d floors is %d \n",
38             n, k, eggDrop(n, k));
39      return 0;
40  }
```

Fig: 5.2.1

5.3 OUTPUT:

```
enter number of eggs and floors : 5 10
Minimum number of trials in worst case with 5 eggs and 10 floors is 4
-----
Process exited after 8.29 seconds with return value 0
Press any key to continue . . .
```

Fig: 5.3.1

6. PROBLEM– 5

LUCKY PALINDROME

In this problem we will convert a given string to a lucky palindrome using minimum number of operations and if several such lucky palindromes are possible, then output the lexicographically smallest one.

6.1 PROBLEM STATEMENT:

The problem is to find the lexicographically smallest palindrome that can be constructed in a greedy manner using minimum number of replacement operations.

This problem can be divided into two tasks:

1. Place substring “lucky” at all possible locations of the string.
2. For each such location of substring “lucky”, find the lexicographically smallest palindrome using minimum number of replacement operations while maintaining the substring “lucky”.

This problem is solvable by greedy algorithm and can also be followed by Brute force approach.

INPUT:

The first line contains a single integer $T \leq 100$ the number of test cases. The following T lines each contain a string of length ≤ 1000 and only containing characters 'a'-'z'.

OUTPUT:

For each line of test case, the program should output on a single line, the required lucky palindrome along with the minimum number of operations, both separated by a single space. If there is no lucky palindrome possible, then just output "unlucky" in a single line.

Sample Input:

Enter the number of test cases: 3

Enter the string: laubcdkey

Enter the string: ytljufcdkuyegy

Enter the string: aaaaa

Sample Output:

luckykcul 8

ygeluckykculegy 9

unlucky

CONCEPTS USED:

1. While loop
2. strlen()
3. strcpy()
4. strcmp()
5. continue
6. for loop
7. if-else if- else statement
8. break

- **While loop:** A while loop in C repeatedly executes a target statement as long as a given condition is true.

Syntax:

```
while(condition)

{

Statement(s);

}
```

- **strlen():** strlen() function takes a string as an argument and returns its length. The returned value is of type size_t(unsigned integer type)

Syntax:

```
int strlen(const char*str)
```

- **strcpy():** The strcpy() function copies the string pointed by source(including the null character) to the destination. The strcpy() function also returns the copied string. It is defined in the string.h header file.

Syntax:

```
Char*strcpy (char*destination,const char*source);
```

- **strcmp():** The strcmp() function compares two strings and returns 0 if both strings are identical. It takes two strings and returns an integer. If the first character of two strings is equal, the next character of two strings are compared. This continues until the

corresponding characters of two strings are different or a null character ‘\0’ is reached. It is defined in string.h header file.

Syntax:

```
int strcmp(const char*str1, const char*str2);
```

- **Continue:** The continue statement in C works somewhat like the break statement. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.

Syntax:

```
Continue;
```

- **For loop:** A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for(initializationStatement ; testExpression ; updateStatement)

{

    // Statements

}
```

- **If- else if -else statement:** The if-else if- else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given conditions true

Syntax:

```
if(Boolean expression)

{

    /*executes when Boolean expression 1 is true*/

}

else if(Boolean expression 2)

{

    /*executes when the boolean expression 2 is true*/

}
```



```

    }

    .....

Else

{

/*executes when the none of the above condition is true*/

}

```

- **Break:**

The break statement is a keyword in C which is used to bring the program control out of the loop the break statement is used inside loops or switch statement. The break statement breaks the loop one by one, i.e, in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break statement in c can be used in the following scenarios:

1. With switch case
2. With loop

Syntax:

```
//loop or switch case
```

```
Break;
```

6.2 CODING:

```
1  #include<stdio.h>
2  #include<string.h>
3  int main(){
4      int t,a,b,i,j,n,min,tm;
5      char s[1001],tmp[1001],ans[1001],luck[6]="lucky";
6      printf("enter the number of test cases : ");
7      scanf("%d",&t);
8      while(t--){
9          printf("enter the string : ");
10         scanf("%s",s);
11         n=strlen(s);
12         if(n<9){
13             printf("unlucky\n");
14             continue;
15         }
16         for(a=0,b=4,min=1001;b<n;a++,b++){
17             strcpy(tmp,s);
18             for(i=a,tm=0;i<=b;i++){
19                 if(tmp[i]!=luck[i-a]){
20                     tmp[i]=luck[i-a];
21                     tm++;
22                 }
23             }
24             for(i=0,j=n-1;i<j;i++,j--){
25                 if(tmp[i]!=tmp[j]){
26                     if(j>=a&j<=b&i>=a&i<=b) break;
27                     if(j>=a&&j<=b) tmp[i]=tmp[j];
28                     else if(i>=a&&i<=b) tmp[j]=tmp[i];
29                     else if(tmp[i]<tmp[j]) tmp[j]=tmp[i];
30                     else tmp[i]=tmp[j];
31                     tm++;
32                 }
33             }
34             if(i>=j){
35                 if(tm<min||tm==min&&strcmp(tmp,ans)<0){
36                     min=tm;
37                     strcpy(ans,tmp);
38                 }
39             }
40             printf("%s %d\n",ans,min);
41         }
42         return 0;
43     }
```

Fig: 6.2.1

6.3 OUTPUT:

```
enter the number of test cases : 3
enter the string : laubcdkey
luckykcul 8
enter the string : ytljufcdekuiegy
ygeluckykculegy 9
enter the string : aaaaaaa
unlucky

-----
Process exited after 46.25 seconds with return value 0
Press any key to continue . . .
```

Fig: 6.3.1

7 SOFTWARE REQUIREMENTS

7.1 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.

7.2 SOFTWARE REQUIREMENTS:

There are two ways to execute this project:

1. Online compilers
2. Softwares for execution (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 user's system specification. These help us to completely execute the project. These softwares are based on the platforms.

8 BIBLIOGRAPHY

1. <https://www.geeksforgeeks.org/count-possible-paths-top-left-bottom-right-nxm-matrix/>
2. <https://bmizepatterson.com/2018/10/01/the-kangaroo-problem/>
3. <https://www.hackerrank.com/challenges/reverse-game/problem>
4. <https://www.tutorialspoint.com/Egg-Dropping-Puzzle>
5. <https://www.geeksforgeeks.org/lexicographically-first-palindromic-string/>