

PROBLEM SOLVING

(Solving various problem using C-language)

*Summer Internship Report Submitted in partial fulfillment
of the requirement for undergraduate degree of*

Bachelor of Technology

In

Computer Science Engineering

By

G Mani Krishna Reddy

221710313014

<https://github.com/Manikrishna14/Problemsolving-221710313014>

Under the Guidance of

Mr.

Assistant Professor



Department Of Computer Science Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

July 2020

DECLARATION

I submitted 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 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

G.MANI KRISHNAREDDY

Date:

221710313014



GITAM (DEEMED TO BE UNIVERSITY)
Hyderabad-502329, India

Dated:

CERTIFICATE

This is to certify that the Industrial Training Report entitled “**Solving various problem using C-language**” is being submitted by G.MANIKRISHNA REDDY(221710313014) 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 2019-20

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

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 completion of this internship.

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

I would like to thank respected **Dr. Phani Kumar**, Head of the Department of Computer Science 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.

G.MANI KRISHNA REDDY
221710313014

TABLE OF CONTENT

1 Introduction to the project	7
2 Problem 1 - String Shifting	8
2.1 Problem Statement	8
2.2 Coding	9
2.3 Output	9
3 Problem 2 - Maximum Row Sum	10
3.1 Problem Statement	10
3.2 Coding	11
3.3 Output	11
4 Problem 3 - Factors	12
4.1 Problem Statement:-	12
4.2 Coding	13
4.3 Output	13
5 Problem 4 - The Prime Numbers	14
5.1 Problem Statement:-	14
5.2 Coding	15
5.3 Output	15
6 Problem 5 - Cryptopangrams	16
6.1 Problem Statement:-	16
6.2 Coding	18
6.3 Output	18
7 Software Requirements	19
7.1 Hardware Requirements	19
7.2 Software Requirements	19
References	20

1 Introduction to the project

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

1. **STRING SHIFTING** -In this problem we are going to print all the possible strings after each rotation(shift).
2. **MAXIMUM ROW SUM**- In this problem we are going to find the row with the maximum sum.
3. **FACTORS** - In this problem we are going to print the factors for the number which is equivalent to the product of the given numbers.
4. **PRIME NUMBERS**- In this problem we are going to print the number of prime numbers in the given range.
5. **CRYPTOGRAMS** - In this problem we recover pangrams based on the inputs.

I have executed projects in C language and Python. For C language, I have used DEV C++ to execute the codes and for Python, I have used Jupyter Notebook and GOOGLE KICKSTART interpreter.

2.PROBLEM 1 STRING SHIFTING

In this problem we are going to print all the possible strings after each rotation(shift).

2.1 Problem Statement:-

Scturtle likes strings very much. He is getting bored today, because he has already completed this week's task and doesn't have anything else to do. So he starts left-rotating a string. If the length of the string is n , then he will rotate it n times and note down the result of each rotation on a paper.

For a string $S=s_1s_2\text{-----}s_n$, n rotations are possible. Let's represent these rotations by $r_1, r_2, \text{-----}, r_n$. Rotating it once will result in $\text{string}r_1=s_2s_3\text{---}s_ns_1$, rotating it again will result in $\text{string}r_2=s_3s_4\text{---}s_ns_1s_2$ and so on. Formally, rotation will be equal to $\text{string}r_i = s_{i+1}s_{i+2}\text{---}s_ns_1\text{---}s_i$. Note that $r_n = s_1s_2\text{---}s_n$.

Your task is to display all n rotations of string S .

For example, if $S = abc$ then it has 3 rotations. They are $r_1 = bca$, $r_2 = cab$ and $r_3 = abc$.

Sample input:-

```
5
abc
abcde
abab
aaa
z
```

Sample output:-

```
bca cab abc
bcdea cdeab deabc eabcd abcde
baba abab baba abab
aaa aaa aaa
z
```

Concepts Used: Concepts Used in this are strings and loops

2.2 Coding:

```
def lr(string):  
    return string[1:len(string)]+string[0]  
tc=int(input())  
for i in range(tc):  
    s=input()  
    for j in range(len(s)):  
        s=lr(s)  
        print(s,end=" ")  
    print("")
```

fig2.2.1

2.3 Output:

Compiled Successfully. memory: 8328

```
bca cab abc  
bcdea cdeab deabc eabcd abcde  
baba abab baba abab  
aaa aaa aaa  
z
```

fig2.3.1

3.PROBLEM 2 THE MAXIMUM ROW SUM

In this problem we are going to find the row with the maximum sum.

3.1 Problem Statement:

Find row with maximum sum in a Matrix

Given an N*N matrix. The task is to find the index of a row with the maximum sum. That is the row whose sum of elements is maximum.

Examples:

Sample Input :

```
1 2 3 4 5
5 3 1 4 2
5 6 7 8 9
0 6 3 4 12
9 7 12 4 3
```

Sample Output :

Row with maximum sum is 3 with sum=35

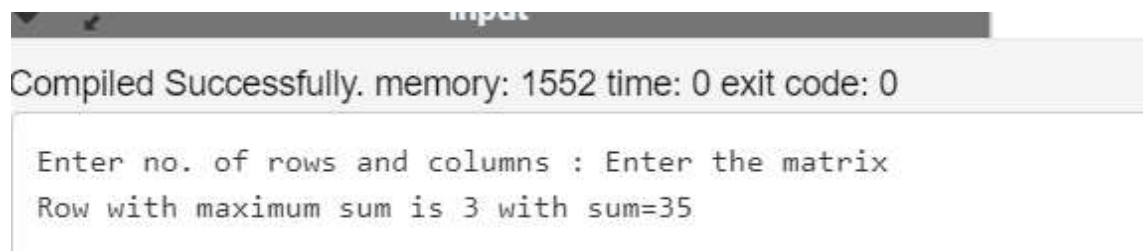
Concepts Used: Concepts Used in this are functions and loops.

3.2 Coding:

```
int i,j,r,c,a[10][10],s;
printf("Enter no. of rows and columns : ");
scanf("%d%d",&r,&c);
printf("Enter the matrix\n");
for(i=0;i<r;i++)
{
    for(j=0;j<c;j++)
        scanf("%d",&a[i][j]);
}
int m=a[0][0],row;
for(i=0;i<r;i++)
{
    s=0;
    for(j=0;j<c;j++)
    {
        s+=a[i][j];
    }
    if(m<s)
    {
        m=s;
        row=i;
    }
}
printf("Row with maximum sum is %d with sum=%d",row+1,m);
}
```

fig3.2.1

3.3 Output:



The screenshot shows a terminal window with the following text:

```
Compiled Successfully. memory: 1552 time: 0 exit code: 0

Enter no. of rows and columns : Enter the matrix
Row with maximum sum is 3 with sum=35
```

fig3.3.1

4.PROBLEM 3 FACTORS

In this problem we are going to print the factors for the number which is equivalent to the product of the given numbers.

4.1 Problem Statement:

Alice has learnt factorization recently. Bob doesn't think she has learnt it properly and hence he has decided to quiz her. Bob gives Alice a very large number and asks her to find out the number of factors of that number. To make it a little easier for her, he represents the number as a product of N numbers. Alice is frightened of big numbers and hence is asking you for help. Your task is simple. Given N numbers, you need to tell the number of distinct factors of the product of these N numbers.

Sample Input:

```
3
3
3 5 7
3
2 4 6
2
5 5
```

Sample Output:

```
8
10
3
```

Concepts Used: Concepts Used in this are loops and lists.

4.2 Coding:

```
for i in range(int(input())):
    n=int(input())
    li=list(map(int,input().split()))
    a=1
    for j in li:
        a*=j
    lii=[]
    for j in range(1,a+1):
        if(a%j==0):
            lii.append(j)
    print(len(lii))
```

fig4.2.1

4.3 Output:

```
3
3
3 5 7
8
2
5 5
3
```

Fig4.3.1

5.PROBLEM 4 THE PRIME NUMBERS

In this problem we are going to print the number of prime numbers in the given range.

5.1 Problem Statement:

Kamal was given a task of identifying the prime numbers between any two numbers and report them accordingly. Help him out by writing a common code which generates prime numbers count between any two given numbers.

Sample Input:

```
2
1 10
3 5
```

Sample Output:

```
4
2
```

Concepts Used: Concepts Used in this are loops and functions.

5.2 Coding:

```
def isprime(n):  
    if(n==1):  
        return False  
    elif(n==2):  
        return True  
    else:  
        f=0  
        for i in range(2,n//2+1):  
            if(n%i==0):  
                f+=1  
        if(f==0):  
            return True  
        else:  
            return False  
for i in range(int(input())):  
    a,b=map(int,input().split())  
    s=0  
    for j in range(a,b+1):  
        if(isprime(j)):  
            s+=1  
    print(s)
```

fig5.2.1

5.3 Output:

```
2  
1 10  
4  
3 5  
2
```

fig5.3.1

6.PROBLEM 5 CRYPTOPANGRAMS

Find the pangrams using deCoding.

6.1 Problem Statement:

On the Code Jam team, we enjoy sending each other *pangrams*, which are phrases that use each letter of the English alphabet at least once. One common example of a pangram is "the quick brown fox jumps over the lazy dog". Sometimes our pangrams contain confidential information — for example, CJ QUIZ: KNOW BEVY OF DP FLUX ALGORITHMS — so we need to keep them secure.

We looked through a cryptography textbook for a few minutes, and we learned that it is very hard to factor products of two large prime numbers, so we devised an encryption scheme based on that fact. First, we made some preparations:

- We chose 26 different prime numbers, none of which is larger than some integer N .
- We sorted those primes in increasing order. Then, we assigned the smallest prime to the letter A, the second smallest prime to the letter B, and so on.
- Everyone on the team memorized this list.

Now, whenever we want to send a pangram as a message, we first remove all spacing to form a plaintext message. Then we write down the product of the prime for the first letter of the plaintext and the prime for the second letter of the plaintext. Then we write down the product of the primes for the second and third plaintext letters, and so on, ending with the product of the primes for the next-to-last and last plaintext letters. This new list of values is our ciphertext. The number of values is one smaller than the number of characters in the plaintext message.

For example, suppose that $N = 103$ and we chose to use the first 26 odd prime numbers, because we worry that it is too easy to factor even numbers. Then $A = 3$, $B = 5$, $C = 7$, $D = 11$, and so on, up to $Z = 103$. Also suppose that we want to encrypt the CJ QUIZ... pangram above, so our plaintext is CJQUIZKNOWBEVYOFDPFLUXALGORITHMS. Then the first value in our ciphertext is 7 (the prime for C) times 31 (the prime for J) = 217; the next value is 1891, and so on, ending with 3053.

We will give you a ciphertext message and the value of N that we used. We will not tell you which primes we used, or how to decrypt the ciphertext. Do you think you can recover the plaintext anyway

Sample Input:

```
2
103 31
217 1891 4819 2291 2987 3811 1739 2491 4717 445 65 1079 8383 5353 901 187 649 1003
697 3239 7663 291 123 779 1007 3551 1943 2117 1679 989 3053
10000 25
3292937 175597 18779 50429 375469 1651121 2102 3722 2376497 611683 489059 2328901
3150061 829981 421301 76409 38477 291931 730241 959821 1664197 3057407 4267589
4729181 5335543
```

Sample Output:

Case #1: CJQUIZKNOWBEVYOFDPFLUXALGORITHMS

Case #2: SUBDERMATOGLYPHICFJKNQVW

Concepts Used: Concepts Used in this are loops and functions.

6.2 Coding:

```
from fractions import gcd
def solve(t):
    input()
    f = list(map(int, input().split(' ')))

    n = len(f)

    message = [-1] * (n + 1)

    st = -1
    for i in range(n-1):
        p = gcd(f[i], f[i+1])
        if p < f[i] and p > 1:
            st = i
            message[i+1] = p
            break
    assert st >= 0

    e = i
    while e >= 0:
        message[e] = f[e] // message[e+1]
        e -= 1
    e = i+2
    while e < len(message):
        message[e] = f[e-1] // message[e-1]
        e += 1

    els = sorted(set(message))
    sol = ""
    for b in range(len(message)):
        for a in range(len(els)):
            if els[a] == message[b]:
                sol += chr(ord('A') + a)
    print("Case #t: %s" % (t, sol))

def main():
    T = int(input())
    for t in range(T):
        solve(t+1)

main()
```

fig6.2.1

6.3Output:

```
Case #1: CJQUIZKNOWBEVYODPFLUXALGORITHMS
Case #2: SUBDERMATOGLYPHICFJKNQVWXZ
```

fig6.3.1

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 projects

- 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

REFERENCES

- <https://www.hackerrank.com/>
- <https://www.geeksforgeeks.org/>
- <https://www.codechef.com/>
- <https://codingcompetitions.withgoogle.com/codejam>