PROBLEM SOLVING

(Solving Various Problems With C Language & C++)

Summer Internship Report Submitted in partial fulfillment

of the requirement for under graduate degree of

Bachelor of Technology

In

Computer Science Engineering

By

Y BUJJI SAI MAHESH

221710313064

https://github.com/saimahesh16

Under the Guidance of

Assistant Professor

Department Of Electronics and Communication 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 With C Language & C++**" 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

Date:

Y Bujji Sai Mahesh

221710313064

GITAM(Deemed To Be University)
Hyderabad-502329
Telangana, India.

Date:

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"Solving Various Problems With C Language & C++"** is being submitted by Y Bujji Sai Mahesh (221710313064) 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.

Mr. ___
Assistant Professor
Department of CSE

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 **Dr.N.Seetharamaiah,** Principal, GITAM Hyderabad

I would like to thank respected **Dr. S. Phani Kumar,** Head of the Department of Electronics and Communication 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.

Y BUJJI SAI MAHESH
221710313064

TABLE OF CONTENTS

1. INTRODUCTION TO THE PROJECT

Problem Solving is the Process of Designing and carrying out the 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. **Saving for a rainy day** - Here, we are going to find out how much a person should deposit today so that he can get assured cash flow as per rate of interest from the given input.
2. **Reverse gear -** Here, we are going to calculate the time taken by the car to hit the wall and we have to park it correctly from the given input.
3. **X total spaces -** Here, we are going to find number of shapes in an string array N*M of O's and the X's and print number of shapes from the given input.
4. **Test vita -** Here, we are going to calculate the minimum number of hours required based on the dependencies from the given input.
5. **Building Blocks** -Here, we are going to calculate the number of blocks that can be piled according to the rules given by their mathematics teacher in the MxN grid from the given input.
6. **Super reduced string** - Here, we are going to reduce the string to its shortest length by performing a series of the operations and print it from the given input.

I have executed projects in C language. For the C language, I have used DEV C++ to execute the codes and questions from the TCS codevita and potential project problems.

2. PROBLEM 1
Saving For a Rainy Day

In this problem we are going to find out how much a person should deposit today so that he can get assured cash flow as per rate of interest.

2.1 Problem 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. Ramu is one such hard working fellow. With a view of future expenses, Ramu resolves to save a certain amount in order to meet his cash flow demands in the future. Consider the following example. Ramu wants to buy a TV. He needs to pay Rs 2000/per month for 12installments to own the TV. If let's say he gets 4% interest per annum on his savings bank account, then Ramu 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 Ramu 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
Sample Input 1
500
3
12
Sample Output 1
1470
Sample Input 2
6000
3
5.9
Sample Output 2
17824

Concepts Used To Solve:-

Computations to get amount to be deposited: we use standard formulas to compute the amount to be deposited

2.2 Coding:-

```
#include<stdio.h>
#include<math.h>
void main()
{
 float m,t,r;
 double a,b;
 int c;
 scanf("%f%f%f",&t,&m,&r);
 b=1200+r*(m-1);
 a=t*m*1200/b;
 c=(int)a;
 printf("the amount to be deposited is :%d\n",c);
}
```

Fig 2.2.1

2.3 OUTPUT:

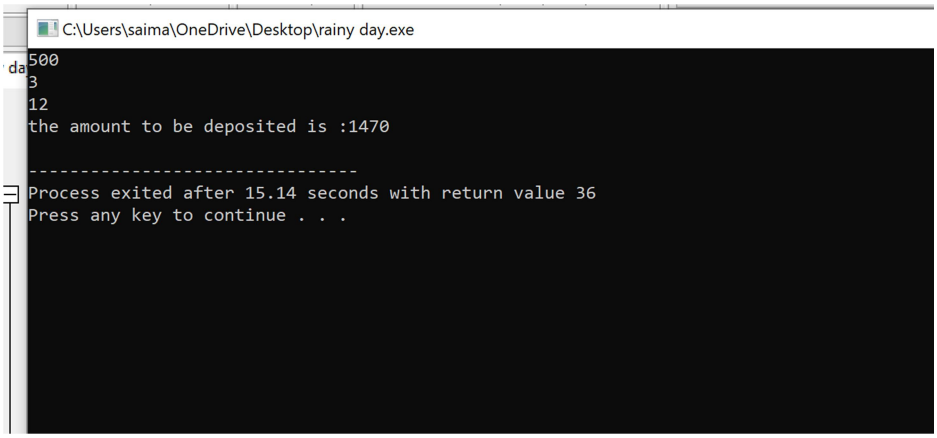

Fig 2.3.1

3. PROBLEM 2
REVERSE GEAR

In this problem we are going to calculate the time taken by the car to hit the wall and we have to park it correctly.

3.1 Problem 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
1. F denotes forward displacement in meters
2. B denotes backward displacement in meters
3. T denotes time taken to cover 1 meter
4. 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
Sample Input and Output

input	Output
2	162
6 9 3 18	220
3 7 5 20	

Concepts Used To Solve:-

Computations , while

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

Syntax:-

```
while(condition) {
   statement(s);
}
```

Flow chart:-

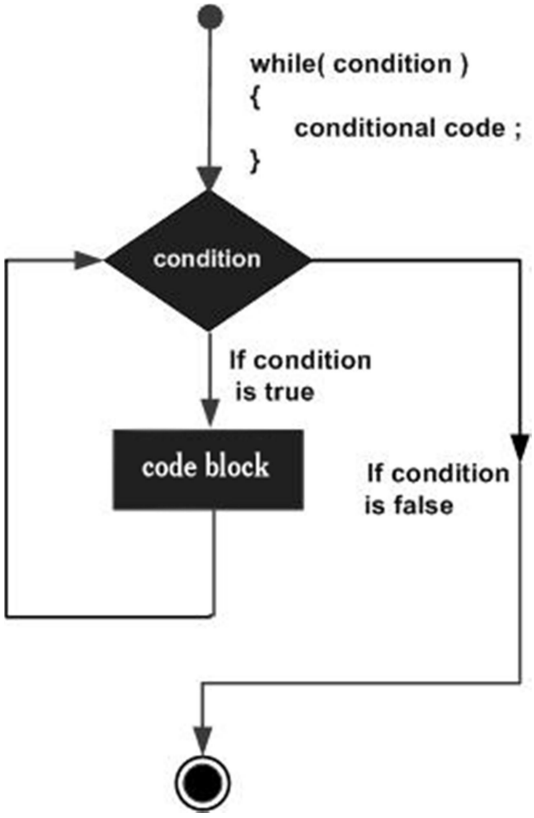

Fig-3.1.1

Computations:-

Computations to get amount to be deposited : we use standard formulas to compute the amount to be deposited

3.2 Coding:-

```c
#include <stdio.h>
int main() {
int f,b,d,t,distance,displacement,step,rd,td,tot,tt,test;
scanf("%d",&test);
while(test)
{
scanf("%d %d %d %d",&f,&b,&t,&d);
distance=b+f;
displacement=b-f;
step=(d-b)/displacement;
if((d-b)%displacement!=0)
step=step+1;
rd=d-(step*displacement);
tot=(step*distance)+rd;
tt=tot*t;
printf("%d\n",tt);
test--;
}
return 0;
```

Fig 3.2.1

3.3 OUTPUT:-

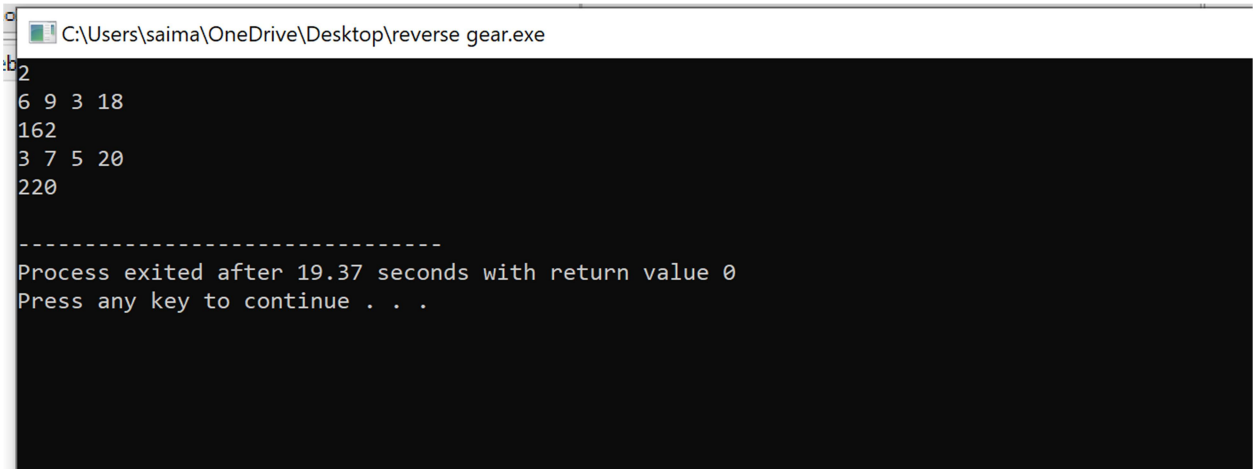

Fig 3.3.1

4. PROBLEM 3
X TOTAL SPACES

In this problem we are going to find number of shapes in an string array N*M of O's and the X's and print number of shapes

4.1 Problem statement:-

Given N * M string array of O's and X's. The task is to find the number of 'X' total shapes. 'X' shape consists of one or more adjacent X's (diagonals not included).

Input:
The first line of input takes the number of test cases T. Then T test cases follow. Each of the T test cases takes 2 input lines. The first line of each test case have two integers N and M.The second line of N space separated strings follow which indicate the elements in the array.

Output:
For each testcase, print number of shapes.

Expected Time Complexity:
O(N*M).
Expected Auxiliary Space:
O(N*M) (recursive).
Example:

Input:

2
4 7
OOOOXXO OXOXOOX XXXXOXO OXXXOOO
10 3
XXO OOX OXO OOO XOX XOX OXO XXO XXX OOO

Output:

4
6
Explanation:
Testcase 1:
Given input is like:

OOOOXXO
OXOXOOX
XXXXOXO
OXXXOOO

So, X with same colour are adjacent to each other vertically for horizontally (diagonals not included). So, there are 4 different groups in the given matrix.

Concepts Used To Solve:-

Graph, DFS

Graph- A graph is a data structure that consists of the following two components:

1. A finite set of vertices also called as nodes.

2. A finite set of ordered pair of the form (u, v) called as edge. Graphs are used to represent many real-life applications: Graphs are used to represent networks.

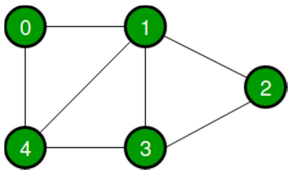

Fig 4.1.1

DFS - Depth First Traversal (or Search) for a graph is similar to Depth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, a node may be visited twice.

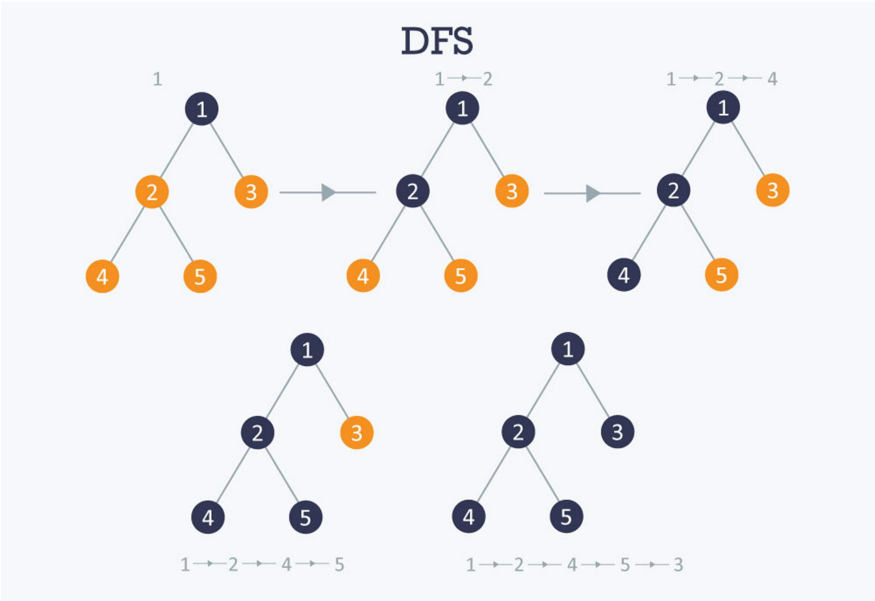

Fig 4.1.2

4.2 Coding :-

```
#include<iostream>
using namespace std;
#include <string.h>
#define MAX 50
int XTotalShapes(char [][MAX],int ,int );
void DFSvisitAll(char [][MAX],int,int,bool[][MAX],int,int);
bool IsConnected(char [][MAX],int,int,bool[][MAX],int,int);
int main()
{
 int t;
 cin>>t;
 while(t--)
 {
  char mat[MAX][MAX];
  int row,col;
  cin>>row>>col;
  for(int i=0;i<row;i++)
   for(int j=0;j<col;j++)
    cin>>mat[i][j];
  cout<<XTotalShapes(mat,row,col)<<endl;
 }
 return 0 ;
}
int XTotalShapes(char mat[][MAX],int row,int col)
{
 bool visited[MAX][MAX];
 memset(visited,0, sizeof(visited));
 int count=0;
 for(int i=0;i<row;i++)
  for(int j=0;j<col;j++)
   if(!visited[i][j]&&mat[i][j]=='X')
```

```
   {
    count++ ;
    DFSvisitAll(mat,row,col,visited,i,j);
   }
 return count;
}
void DFSvisitAll(char mat[][MAX],int row,int col,bool visited[][MAX],int currR,int currC)
{
 visited[currR][currC]=true;
 int t1[]={-1,0,0,1};
 int t2[]={0,-1,1,0};
 for(int k=0;k<4;k++){
 if(IsConnected(mat,row,col,visited,currR+t1[k],currC+t2[k]))
 DFSvisitAll(mat,row,col,visited,currR+t1[k],currC+t2[k]) ;
 }
}
bool IsConnected(char mat[][MAX],int row,int col,bool visited[][MAX],int currR,int currC)
{
 return (currR>=0&&currC>=0)&&(currR<row&&currC<col)&&(mat[currR][currC]=='X'&&!visited[currR][currC]) ;
}
```

Fig 4.2.1

4.3 OUTPUT :-

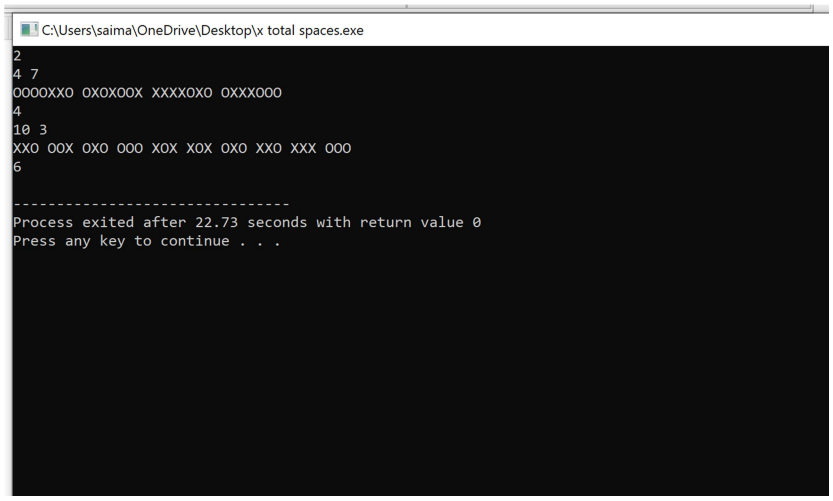

Fig 4.3.1

5. PROBLEM 4
TESTVITA

In this problem, we are going to calculate the minimum number of hours required based on the dependencies.

5.1 Problem statement:-

TCS is working on a new project called "TestVita". There are N modules in the project. Each module (i) has completion time denoted in number of hours (Hi) and may depend on other modules. If Module x depends on Module y then one needs to complete y before x.
As Project manager, you are asked to deliver the project as early as possible. Provide an estimation of amount of time required to complete the project.

Input Format:

First line contains T, number of test cases.
For each test case:
1. First line contains N, number of modules.
2. Next N lines, each contain:
(i) Module ID
(Hi) Number of hours it takes to complete the module
(D) Set of module ids that i depends on integers delimited by space.

Output Format:

Output the minimum number of hours required to deliver the project.
Constraints:
1. 1 <= T <= 10
2. 0 < N < 1000; number of modules
3. 0 < i <= N; module ID
4. 0 < Hi < 60; number of hours it takes to complete the module i
5. 0 <= |D| < N; number of dependencies
6. 0 < Dk <= N; module ID of dependencies
Sample Input
1
5
1 5
2 6 1
3 3 2

4 2 3
5 1 3
Sample Output
16

Explanation:

Module 2 depends on module 1, hence complete module 1 first
After completing module 1 we can complete module 2 and then module 3
Module 4 and 5 can be started simultaneously in parallel after module 3 is completed.
Hence the answer = 5 + 6 + 3 + 2 = 16

Concepts Used To Solve:-

Loops, Array

Loops - A for loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line
.

Syntax:-

```
for ( init; condition; increment ) {
   statement(s);
}
```

Arrays - An array in C or C++ is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They are used to store similar type of elements as in the data type must be the same for all elements. They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type.

Syntax -

```
Datatype arrayName [ arraySize ];
```

5.2 Coding:-

```
#include <stdio.h>
int main() {
 int n,b[10],t[10],c[10],q,a,i,sum=0;
 scanf("%d",&a);
 for(q=1;q<=a;q++)
 {
 scanf("%d",&n);
 for(i=0;i<n;i++)
    scanf("%d %d %d",&b[i],&t[i],&c[i]);
    for(i=0;i<n-1;i++)
    {
            if(c[i]==c[i+1])
            {
                if(t[i]<t[i+1])
                t[i]=0;
                else
                t[i+1]=0;
            }
    }
   for(i=0;i<n;i++)
   {
       sum=sum+t[i];
   }
    printf("%d",sum);
 }
 return 0;
}
```

Fig 5.2.1

5.3 OUTPUT :-

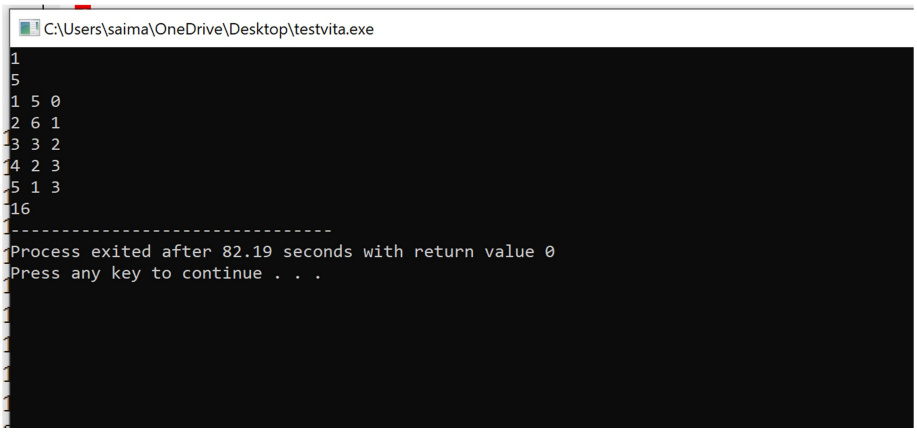

Fig 5.3.1

6. PROBLEM 5
BUILDING BLOCKS

In this problem we are going to calculate number of blocks that can be piled according to the rules given by their mathematics teacher in the MxN grid

6.1 Problem Statement:-

The Mathematics teacher wanted to introduce a new competition to the students to sharpen their skills in optimization. He drew a rectangular M ×N grid on the ground and filled it with some non-negative integers on each cell of the grid. The cell named (i,j) is at the intersection of i th row and j th column. He gave the following challenge to the students:
1. On each cell of the grid, you can pile any number of cube blocks.
2. Each layer must be rectangular (with no gaps) and rest p supported completely by the immediate below layer.
3. The number of blocks on each cell should not exceed the number written on the cell on the ground.
4. In each layer, the cell above (1,1) must be covered.
The challenge is to pile up the maximum number of blocks subject to the above conditions.
For example if the bottom grid was as follows:
1 1 0
1 1 1
1 1 0the maximum number of blocks you can pile is 6 with one layer covering all the cells from (1,1) to (3,2).
Constraints
1 <= M,N <= 50
Maximum value in each cell is 50

Input Format:

The first line will contain two comma separated integers M,N giving the size of the grid, where M is the number of rows and N is the number of columns.
The next M lines will each contain comma separated N non-negative integers giving the numbers in the grid cells.

Output:

One line containing the number of blocks that can be piled according to the rules.

Explanation:

Example 1:
Input:

3,4
5,4,9,3
4,3,5,6
2,2,1,1
Output:

32

Explanation:

One example of the maximum number of blocks that could be piled on the grid is shown below:

5	4	4	3
3	3	3	3
1	1	1	1

The total number of blocks is 32. Hence the output is 32.

Example 2:
Input:

4,7
27,26,28,14,15,38,0
38,40,35,2,20,43,39
18,48,43,2,47,18,26
38,2,29,23,14,31,32

Output:
242
Explanation
The number of rows is 4. The number of columns is 7. The values in the cells of the grid are

27	26	28	14	15	38	0
38	40	35	2	20	43	39
18	48	43	2	47	18	26
38	2	29	23	14	31	32

One possible maximal placing of the blocks is

27	26	26	2	2	2	0
27	26	26	2	2	2	0
18	18	18	2	2	2	0

2 2 2 2 2 2 0
As there are 242 blocks in this maximal placing, the output is 242.

Concepts Used To Solve:-

Matrices , For

Matrices-

A matrix is a collection of elements of the same type, organized in the form of a table. Each element is indexed by a pair of numbers that identify the row and the column of the element. A matrix can be represented in Java through an array, whose elements are themselves (references to) arrays representing the various rows of the matrix.

Declaration of a matrix:-

A matrix is declared in the following way (as an array of arrays): int[][] m; // declaration of an array of arrays (matrix) m

For -A for loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line.

Syntax:-

```
for ( init; condition; increment ) {
   statement(s);
}
```

6.2 Coding:-

```
#include<stdio.h>
int m[10][10];
void del(int i,int j,int s,int f,int r,int c)
{
    int k,l;
    if(s==0)
    {
        for(k=0;k<r;k++)
            for(l=j;l<c;l++)
                m[k][l]=f-1;
    }
    else
    {
        for(k=i;k<r;k++)
            for(l=0;l<c;l++)
                m[k][l]=f-1;
    }
}
int rsum(int i,int j,int r,int c)
{
    int k,l,s=0;
    for(k=0;k<i;k++)
        for(l=0;l<c;l++)
            s+=m[k][l];
    return s;
}
int csum(int i,int j,int r,int c)
{
    int k,l,s=0;
    for(k=0;k<r;k++)
        for(l=0;l<j;l++)
```

```
            s+=m[k][l];
    return s;
}
int main()
{
    int i,j,k,r,c,max=0,r1,c1,s,t1,t2,sum=0;
    char ch;
    scanf("%d%c%d",&r,&ch,&c);
    r1=r;
    c1=c;
    for(i=0;i<r;i++)
        for(j=0;j<c;j++)
        {
            scanf("%d",&m[i][j]);
            if(j<c-1)
            scanf("%c",&ch);
            if(m[i][j] > max)
                max=m[i][j];
        }
    for(k=1;k<=max;k++)
    {
        for(i=0;i<r1;i++)
            for(j=0;j<c1;j++)
            {
                if(m[i][j] < k)
                {
                    t1=rsum(i,j,r1,c1);
                    t2=csum(i,j,r1,c1);
                    s=(t1 > t2) ? 1 : 0;
                    del(i,j,s,k,r1,c1);
                    if(s==1)
                        r1=i;
                    else
                        c1=j;
                }
            }
    }
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        sum+=m[i][j];
    }
    printf("%d",sum);
    return 0;
}
```

Fig 6.2.1

6.3 OUTPUT :-

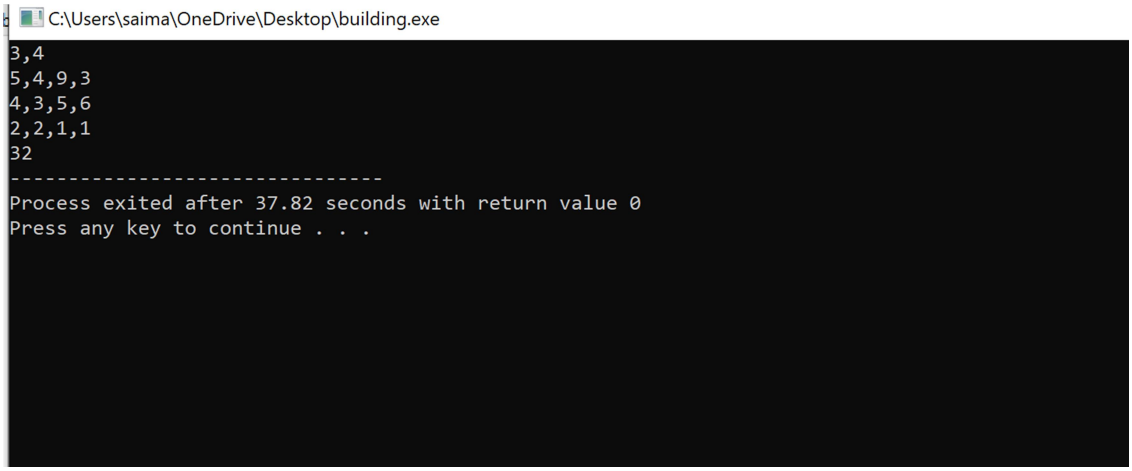

Fig 6.3.1

7. PROBLEM 6
SUPER REDUCED STRING

In this problem we are going to reduce the string to its shortest length by performing series of the operations and print it

7.1 Problem Statement:-

Steve has a string of lowercase characters in range ascii[‘a’..’z’]. He wants to reduce the string to its shortest length by doing a series of operations. In each operation he selects a pair of adjacent lowercase letters that match, and he deletes them. For instance, the string aab could be shortened to b in one operation. Steve’s task is to delete as many characters as possible using this method and print the resulting string. If the final string is empty, print Empty String Function Description Complete the superReducedString function in the editor below. It should return the super reduced string or Empty String if the final string is empty.
superReducedString has the following parameter(s):
s: a string to reduce

Input Format:

A single string,s.

Output Format:

If the final string is empty, print Empty String; otherwise, print the final non-reducible string.

Sample Input 0
aaabccddd
Sample Output 0
abd
Explanation 0
Steve performs the following sequence of operations to get the final string:
aaabccddd → abccddd → abddd → abd
Sample Input 1
aa
Sample Output 1
Empty String
Explanation 1
aa → Empty String
Sample Input 2

baab
Sample Output 2
Empty String
Explanation 2
baab → bb → Empty String

Concepts Used To Solve:-

Character Array , Empty function()

Character Array - Character (char) is a primitive data type in C that is used to store a character value.
A string is a sequence of characters treated as a single entity and is terminated by a null character ('\0').
Example: char arr[] = "Hello World";
Empty function() - This function checks whether the string is empty or not. Function returns a Boolean value either true or false.

Syntax

Consider a string str. Syntax would be:

str.empty();

Parameter

This function does not contain any parameter.

Return value

It returns a Boolean value either 0 or 1 depending upon the conditions

7.2 Coding:-

```
#include<iostream>
using namespace std;
int main()
{
    string str;
    cin >> str;
    int len=str.length(),i=0;
    while(i<len)
    {
        if(str[i]==str[i+1]&&i>-1)
        {
            str.erase(i,2);
            i--;
        }
        else
        i++;
 }
if(!str.empty())
cout<<str<<endl;
else
cout<<"Empty String"<<endl;
return 0;
}
```

Fig 7.2.1

7.3 OUTPUT:-

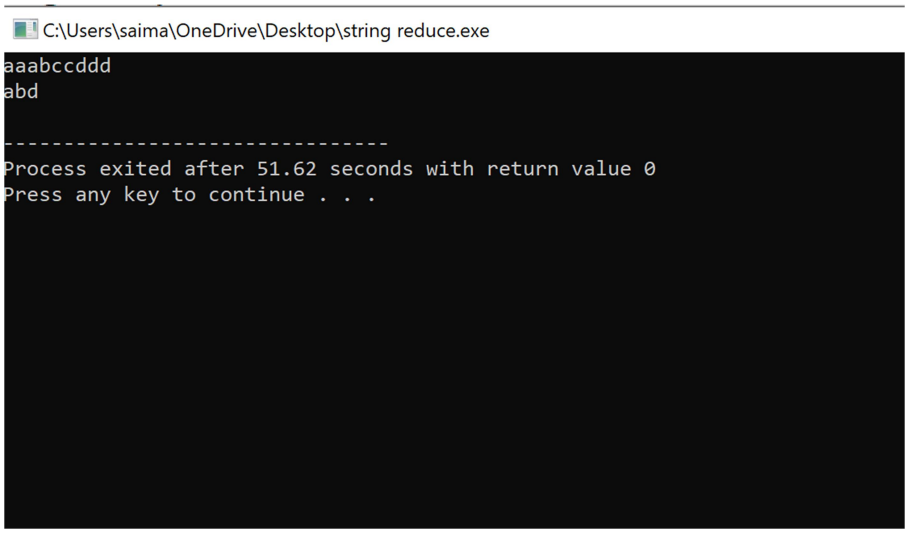

Fig 7.3.1

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 applications for execution..,IDE'S(DEV C++)

Online compilers are tools which allows you to compile source code and execute it online in many programming languages. It's an online compiler and debugging tool which allows to compile and run code online

IDEs 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. Many of them are free of cost.

Based on the availability we can use anyone.

Some of the best Online Compilers are:
OnlineGDB, Tutorial point, Programiz, Code chef and many more.

Some of the best Softwares for execution(IDE's) are:
For C: Turbo C, DEV C++ etc.

Dev-C++ is a free full-featured integrated development environment (IDE) distributed under the GNU General Public License for programming in C and C++.

REFERENCES

- https://www.programminggeek.in
- https://practice.geeksforgeeks.org
- www.tcscodevita.com
- https://www.hackerrank.com