Experiment No. 11

****

Title: Minimum spanning tree algorithm

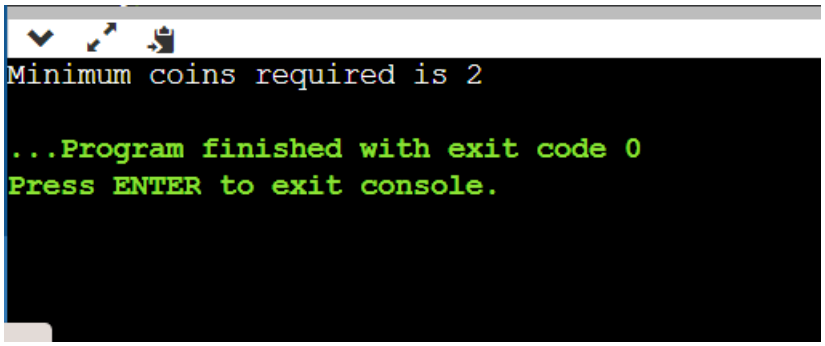
Name: : Rahulkumar Varma

Class: TE CSE **Batch:** T1 **Roll No:** 3020

```
#include <limits.h>
//Library for creating the set
#include <stdbool.h>
#include <stdio.h>
#define Vertices 5
//Finding the vertex with the lowest key value from a set of vertices not included in MST
int Least_Key(int key[], bool Min_Span_Tree[])
{
   int least = INT_MAX, min_index;
   for (int v = 0; v < Vertices; v++)
      if (Min_Span_Tree[v] == false && key[v] < least)
         least = key[v], min_index = v;
   return min_index;
}
//Utility function to print MST
int print_Prims_MST(int parent[], int graph[Vertices][Vertices])
{
   printf("Edge \tWeight\n");
   for (int i = 1; i < Vertices; i++)
      printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
}
//Function for generating an MST
void prims_MST(int graph[Vertices][Vertices])
{
   int parent[Vertices];
   int key[Vertices];
   bool Min_Span_Tree[Vertices];
   for (int i = 0; i < Vertices; i++)
      key[i] = INT_MAX, Min_Span_Tree[i] = false;
   key[0] = 0;
```

```
    parent[0] = -1;
    for (int count = 0; count < Vertices - 1; count++) {
        int u = Least_Key(key, Min_Span_Tree);
        Min_Span_Tree[u] = true;
        for (int v = 0; v < Vertices; v++)
            if (graph[u][v] && Min_Span_Tree[v] == false && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
    printf("Created Spanning Tree for Given Graph is: \n");
    printf("\n");
    print_Prims_MST(parent, graph);
}
//Driver method
int main()
{
    int graph[Vertices][Vertices] = { { 0, 3, 2, 0, 0 },
                    { 3, 0, 16, 12, 0 },
                    { 2, 16, 0, 0, 5 },
                    { 0, 12, 0, 0, 0 },
                    { 0, 0, 5, 0, 0 } };
    prims_MST(graph);
    return 0;
}
```

OUTPUT



Experiment No. 12

****

Title: Finding shortest path using Dijkstra's algorithm

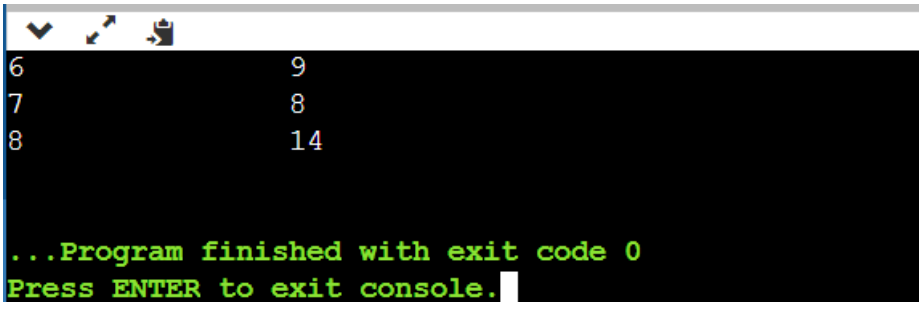
Name: : Rahulkumar Varma

Class: TE CSE **Batch:** T1 **Roll No:** 3020

```
#include <iostream>
using namespace std;
#include <limits.h>
int minDistance(int dist[], bool sptSet[])
{
	// Initialize min value
	int min = INT_MAX, min_index;

	for (int v = 0; v < V; v++)
		if (sptSet[v] == false && dist[v] <= min)
			min = dist[v], min_index = v;
	return min_index;
}
// A utility function to print the constructed distance array
void printSolution(int dist[])
{
	cout <<"Vertex \t Distance from Source" << endl;
	for (int i = 0; i < V; i++)
		cout << i << " \t\t"<<dist[i]<< endl;
}
// Function that implements Dijkstra's single source shortest path algorithm
// for a graph represented using adjacency matrix representation
void dijkstra(int graph[V][V], int src)
{
	int dist[V]; // The output array. dist[i] will hold the shortest
	// distance from src to i

	bool sptSet[V]; // sptSet[i] will be true if vertex i is included in shortest
	// path tree or shortest distance from src to i is finalized
```

```
        // Initialize all distances as INFINITE and stpSet[] as false
        for (int i = 0; i < V; i++)
                dist[i] = INT_MAX, sptSet[i] = false;
        // Distance of source vertex from itself is always 0
        dist[src] = 0;
                for (int count = 0; count < V - 1; count++) {
                int u = minDistance(dist, sptSet);
                sptSet[u] = true;
                for (int v = 0; v < V; v++)
                        if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                                && dist[u] + graph[u][v] < dist[v])
                                dist[v] = dist[u] + graph[u][v];
        }
        printSolution(dist);
}
int main()
{
        /* Let us create the example graph discussed above */
        int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
                                                { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
                                                { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
                                                { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
                                                { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
                                                { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
                                                { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
                                                { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
                                                { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };
        dijkstra(graph, 0);
        return 0;
}
```

OUTPUT-



Experiment No. 13

****

Title: Experiment on jolly jumper sequencing.

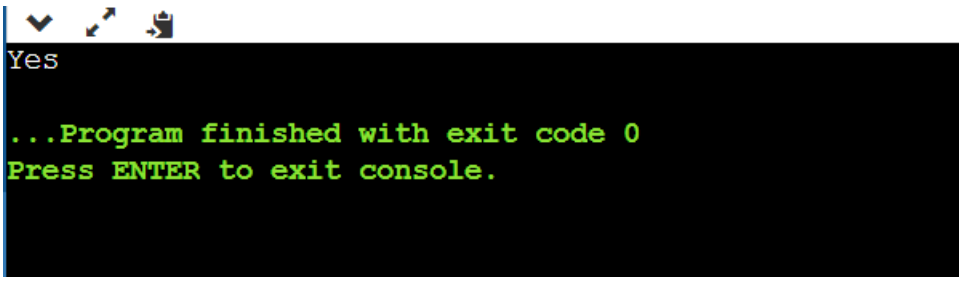
Name: : Rahulkumar Varma

Class: TE CSE **Batch:** T1 **Roll No:** 3020

```
// Program for Jolly Jumper Sequence
#include<bits/stdc++.h>
using namespace std;

// Function to check whether given sequence is
// Jolly Jumper or not
bool isJolly(int a[], int n)
{
	// Boolean vector to diffSet set of differences.
	// The vector is initialized as false.
	vector<bool> diffSet(n, false);

	// Traverse all array elements
	for (int i=0; i < n-1 ; i++)
	{
		// Find absolute difference between current two
		int d = abs(a[i]-a[i+1]);

		// If difference is out of range or repeated,
		// return false.
		if (d == 0 || d > n-1 || diffSet[d] == true)
			return false;

		// Set presence of d in set.
		diffSet[d] = true;
	}

	return true;
}

// Driver Code
int main()
```

```
{
        int a[] = {11, 7, 4, 2, 1, 6};
        int n = sizeof(a)/ sizeof(a[0]);
        isJolly(a, n)? cout << "Yes" : cout << "No";
        return 0;
}
```

OUTPUT-



Experiment No. 14

****

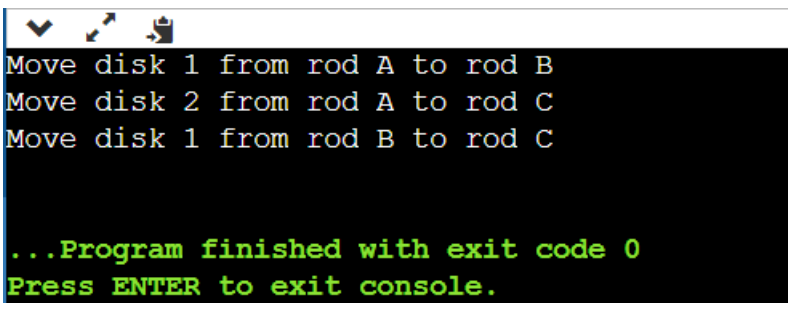
Title: Experiment on tower Hanoi.

Name: : Rahulkumar Varma

Class: TE CSE **Batch:** T1 **Roll No:** 3020

```
#include <bits/stdc++.h>
using namespace std;
void towerOfHanoi(int n, char from_rod,
                                        char to_rod, char aux_rod)
{
        if (n == 0)
        {
                return;
        }
        towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
        cout << "Move disk " << n << " from rod " << from_rod <<
                                                        " to rod " << to_rod << endl;
        towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
// Driver code
int main()
{
        int n = 4; // Number of disks
        towerOfHanoi(n, 'A', 'C', 'B'); // A, B and C are names of rods
        return 0;
}
```

OUTPUT



Experiment no. 15

Title-Program to check whether the input graph is bicolor or not.

Name :- Rahulkumar Varma Class :- TY CSE

Roll no. :- 3020 Batch :- T1

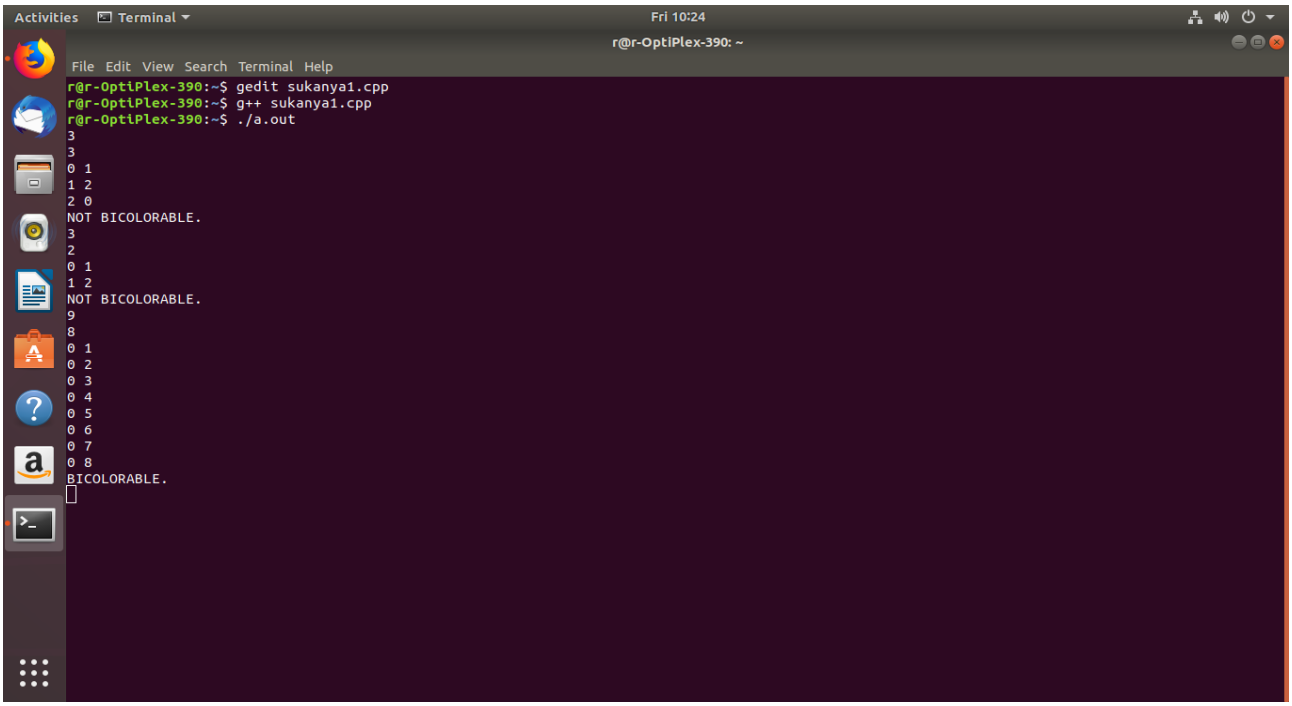
****

PROGRAM :-

```
#include<iostream>
#include<stdio.h>
#include<queue>
#define SIZE 201
#define YES -1
using namespace std;
int main()
{
   struct graph{int connect;}G[200][200];
   int visit[200],color[200];
   queue<int> Q;
int n,l,i,x,y,check,u,j;
while( scanf("%d",&n) == 1 && n != 0 )
{
          scanf("%d",&l);
           for(i=0;i<n;i++)                // Graph initialisation
                                    {
                                    color[i] = NO;            // initially all nodes not colored
                                visit[i] = NO;
                                     for(j=0;j<n;j++)
                                        G[i][j].connect  = NO;
                                    }
for(i=0;i<l;i++)
{
scanf("%d %d",&x,&y);
G[x][y].connect = YES;
G[y][x].connect = YES;
}

check = NO;
```

```
color[0] = 0;                          // coloring 0th node as 0
Q.push(0);
visit[0] = YES;
while(Q.size()>0 && check==NO)
{

   u  = Q.front();
   Q.pop();
   for(i=0;i<n && check==NO ;i++)
      if(G[u][i].connect==YES && (u!=i) )
      {
      if(visit[i]==NO )
         {
            visit[i] = YES;
             if(color[i]==NO)
             {
             if(color[u]==0)
             color[i] = 1;
             else
             color[i] = 0;}

                 Q.push(i);
               }
       else if(color[i]==color[u])
        {
         printf("NOT BICOLORABLE.\n");
         check = YES;}              // check is basically used to break out of loop
            }

}
if(check == NO)
printf("BICOLORABLE.\n");
}
return 0;
}
```

OUTPUT :-


Activities
Terminal
Fri 10:24
r@r-OptiPlex-390: ~
File Edit View Search Terminal Help
r@r-OptiPlex-390:~$ gedit sukanya1.cpp
r@r-OptiPlex-390:~$ g++ sukanya1.cpp
r@r-OptiPlex-390:~$ ./a.out
3
3
0 1
1 2
2 0
NOT BICOLORABLE.
3
2
0 1
1 2
NOT BICOLORABLE.
9
8
0 1
0 2
0 3
0 4
0 5
0 6
0 7
0 8
BICOLORABLE.

Experiment no. 16

Title-Expiriment on problem is Bigger and smarter.

Name :- Rahulkumar Varma Class :- TY CSE

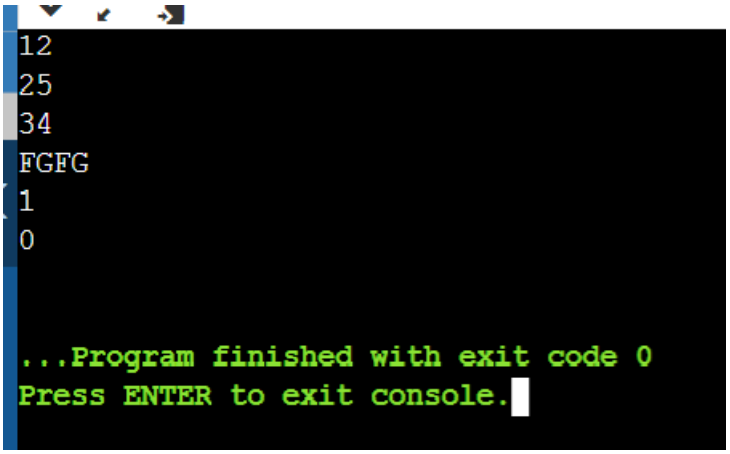
Roll no. :- 3020 Batch :- T1

PROGRAM :-

```
#include <cstdio>
#include <algorithm>
using namespace std;
struct E{
   int ind,W,S;

   E(){
   }

   E(int _ind, int _W, int _S){
      ind=_ind;
      W=_W;
      S=_S;
   }

   bool operator < (E X) const{
      if(W!=X.W) return W<X.W;
      return S>X.S;
   }
};

int main(){
   int n=0,W,S;
   E a[1000];

   while(scanf("%d %d",&W,&S)==2) a[n]=E(++n,W,S);
   sort(a,a+n);

   int dp[n],next[n],ans=0,start;
```

```
    for(int i=n-1;i>=0;i--){
        dp[i]=1;
        next[i]=-1;
        for(int j=i+1;j<n;j++){
            if(a[i].W<a[j].W && a[i].S>a[j].S && 1+dp[j]>dp[i]){
                dp[i]=1+dp[j];
                next[i]=j;
            }
        }
        if(dp[i]>ans){
            ans=dp[i];
            start=i;
        }
    }
    printf("%d\n",ans);
    for(int i=start;i!=-1;i=next[i]) printf("%d\n",a[i].ind);

    return 0;
}
```

Output :



Experiment no. 17

Title-Implement edit step ladder problem.

Name :- Rahulkumar Varma Class :- TY CSE

Roll no. :- 3020 Batch :- T1

****

PROGRAM :-

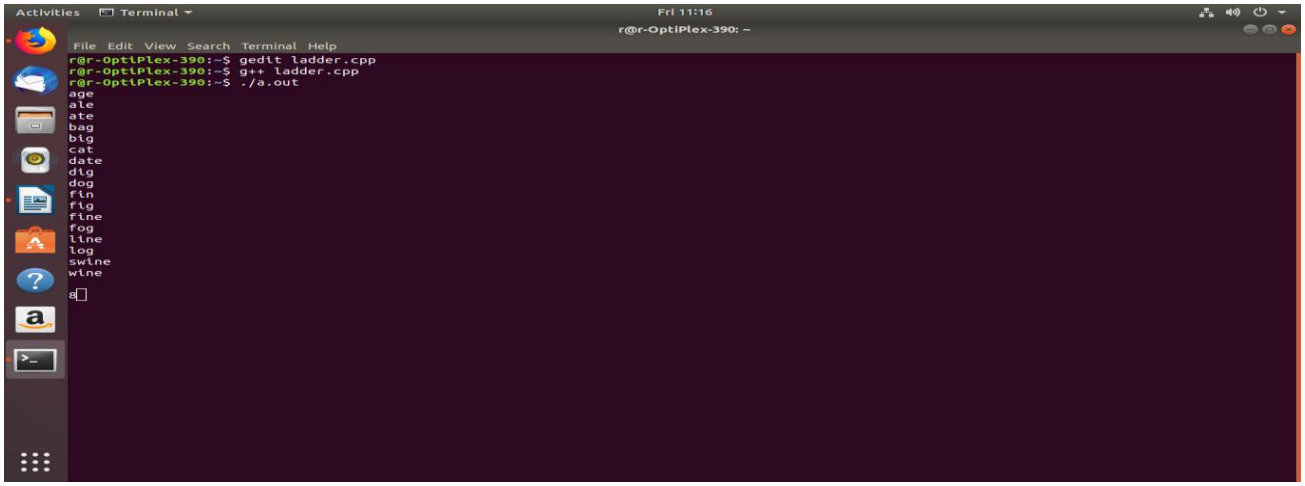
```
#include<iostream>
#include <vector>
#include <string>
#include <map>
using namespace std;

int LevDistance(string a,string b,int m,int n){

  int d[m+1][n+1],minimum;
  int x;
  int y;
  int z;

  for(int i=0;i<=m;i++){
    d[i][0]=i;
  }
  for(int i=0;i<=n;i++){
    d[0][i]=i;
  }

  for(int i=1; i<=m;i++){
    for(int j=1;j<=n;j++){
      if(a[i-1]==b[j-1]){
        d[i][j]=d[i-1][j-1];
      }
      else{

        x = d[i-1][j]+1;
        y = d[i][j-1]+1;
```

```
        z = d[i-1][j-1]+1;

        if(x < y && x < z){
         minimum = x;
        }
        else if(y <x && y<z){
           minimum = y;
        }
        else{
           minimum = z;
        }
        d[i][j] = minimum
          }
    }
  }


  return d[m][n];

}

int steps(map<string, vector <string> > links, int wordcount,string word){

  int newwordcount;
  int words=wordcount;
  for(int j=0;j<links[word].size();j++){


    newwordcount = steps(links,wordcount+1,links[word][j]);
    //    cout << newwordcount << " " << word << endl;
    if(newwordcount > words){
      words=newwordcount;
    }
  }
  return words;

}

int main(){
  map<string,vector <string> > links;
```

```
 vector<string> words;
 string word, word2;
 int wordcount,wordcounts=0;

 while(cin >> word){


  words.push_back(word);
  if(cin.eof()){
    break;
  }

 }
 int minimum;
 //  cout << words[5];
 for(int k=0;k<words.size();k++){
  //cout << words[k] << endl;
  for(int j=k+1;j<words.size();j++){
    //cout << words[j] << endl;
      minimum=LevDistance(words[k],words[j],words[k].size(),words[j].size());
if(minimum==1){
//  cout << words[j] << " " << words[k] << endl;
        links[words[k]].push_back(words[j]);
      }

  }

  }

  for(int k=0;k<words.size();k++){
    //   cout << words[k] << endl;
    for(int j=0;j<links[words[k]].size();j++){
      //cout << links[words[k]][j];
    }
    //cout << endl;
    //cout <<endl;
  }

  for(int k=0; k<words.size(); k++){
```

```
      wordcount = steps(links,1,words[k]);
      //cout << wordcount << endl;
      if(wordcount > wordcounts){
        wordcounts = wordcount;
      }
    }

    cout << wordcounts<< endl;
  return 0;
}
```

OUTPUT :



Experiment no. 18

Title-Implement distinct subsequent problem.

Name :- Rahulkumar Varma Class :- TY CSE

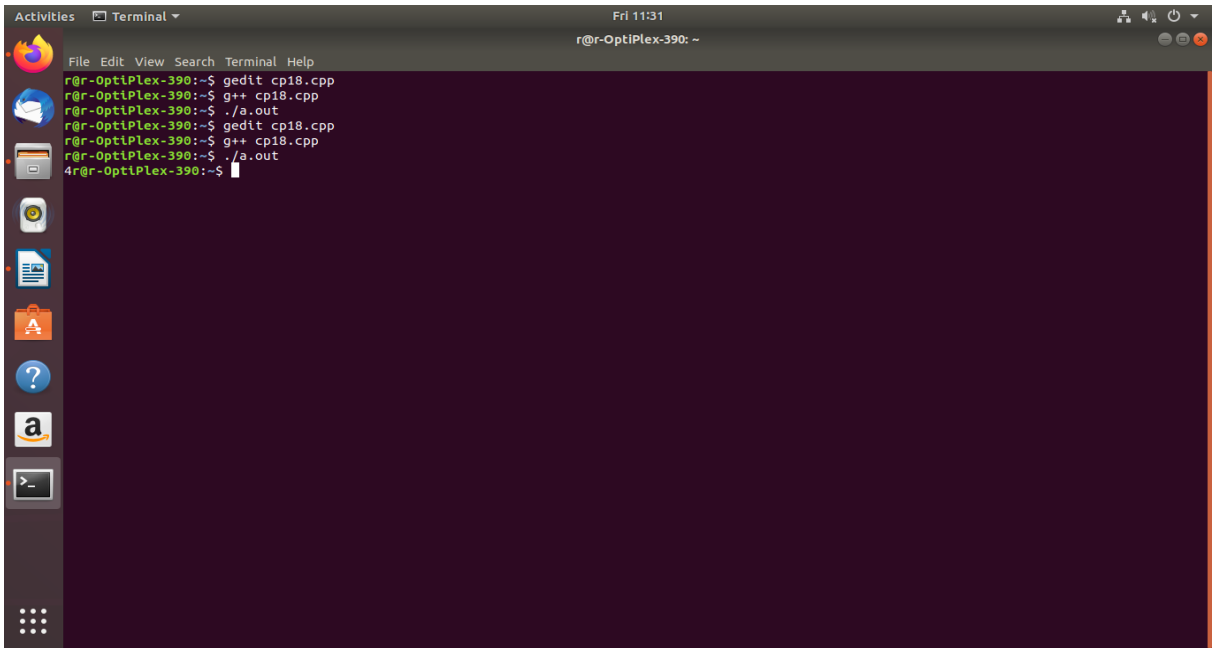
Roll no. :- 3020 Batch :- T1

PROGRAM :-

```
#include <bits/stdc++.h>
using namespace std;

// Create an empty set to store the subsequences
unordered_set<string> sn;

// Function for generating the subsequences
void subsequences(char s[], char op[], int i, int j)
{

	// Base Case
	if (s[i] == '\0') {
		op[j] = '\0';

		// Insert each generated
		// subsequence into the set
		sn.insert(op);
		return;
	}

	// Recursive Case
	else {
		// When a particular character is taken
		op[j] = s[i];
		subsequences(s, op, i + 1, j + 1);

		// When a particular character isn't taken
		subsequences(s, op, i + 1, j);
		return;
	}
}
```

```
// Driver Code
int main()
{
        char str[] = "ggg";
        int m = sizeof(str) / sizeof(char);
        int n = pow(2, m) + 1;

        // Output array for storing
        // the generating subsequences
        // in each call
        char op[m+1]; //extra one for having \0 at the end

        // Function Call
        subsequences(str, op, 0, 0);

        // Output will be the number
        // of elements in the set
        cout << sn.size();
        sn.clear();
        return 0;

        // This code is contributed by Kishan Mishra
}
```

OUTPUT :-



Experiment no. 19

Title-Implement weight and measure problem.

Name :- Rahulkumar Varma Class :- TY CSE

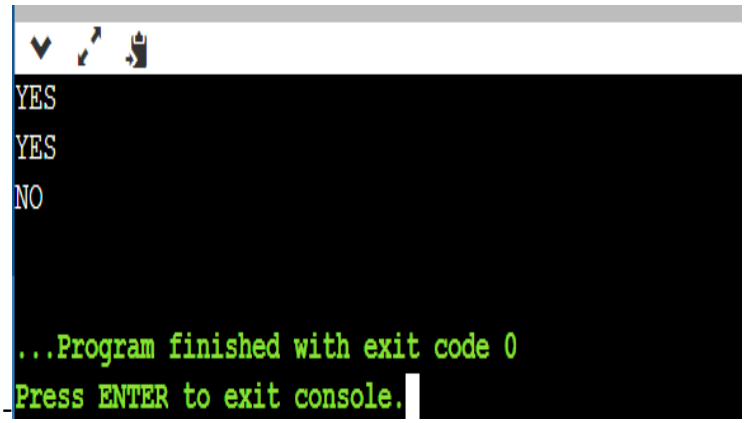
Roll no. :- 3020 Batch :- T1

```
#include <bits/stdc++.h>
using namespace std;

int found = 0;

void solve(int idx, int itemWt, int wts[],
           int N)
{
    if (found)
        return;

    // Item has been measured
    if (itemWt == 0) {
        found = 1;
        return;
    }


    if (idx > N)
        return;


    solve(idx + 1, itemWt, wts, N);


    solve(idx + 1, itemWt + wts[idx], wts,
        N);


    solve(idx + 1, itemWt - wts[idx], wts,
        N);
}
```

```
bool checkItem(int a, int W)
{
	// If the a is 2 or 3, answer always
	// exists
	if (a == 2 || a == 3)
		return 1;

	int wts[100]; // weights array
	int totalWts = 0; // feasible weights
	wts[0] = 1;
	for (int i = 1;; i++) {
		wts[i] = wts[i - 1] * a;
		totalWts++;

		// if the current weight
		// becomes greater than 1e9
		// break from the loop
		if (wts[i] > 1e9)
			break;
	}
	solve(0, W, wts, totalWts);
	if (found)
		return 1;

	// Item can't be measured
	return 0;
}

int main()
{
	int a = 2, W = 5;
	if (checkItem(a, W))
		cout << "YES" << endl;
	else
		cout << "NO" << endl;

	a = 4, W = 11, found = 0;
	if (checkItem(a, W))
		cout << "YES" << endl;
```

```
        else
                cout << "NO" << endl;

        a = 4, W = 7, found = 0;
        if (checkItem(a, W))
                cout << "YES" << endl;
        else
                cout << "NO" << endl;
        return 0;
}
```

Output-



Experiment no. 20

Title-Implement chopstick problem.

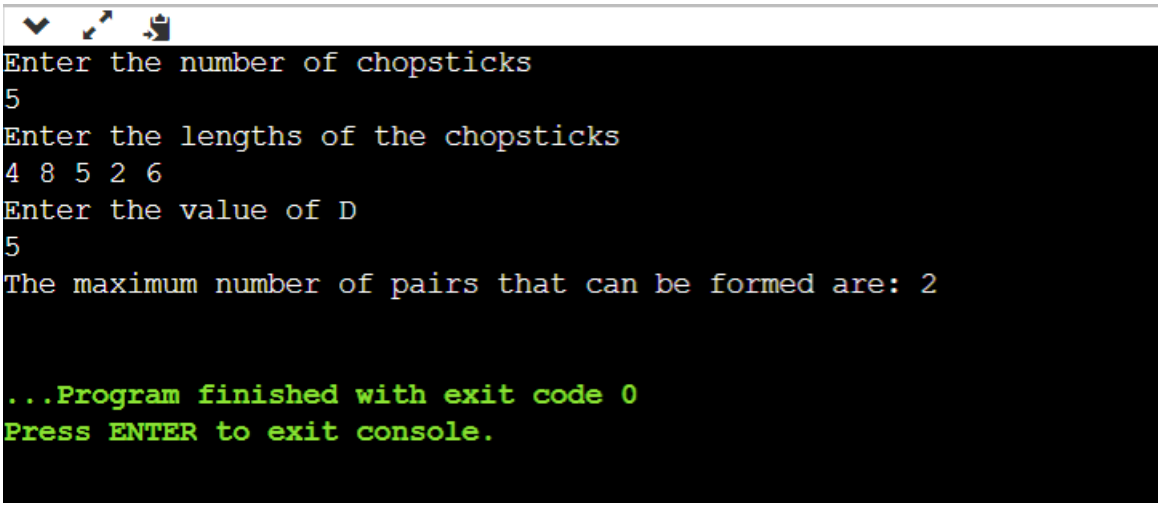
Name :- Rahulkumar Varma Class :- TY CSE

Roll no. :- 3020 Batch :- T1

```
#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

int max_chopsticks_pairs(vector <int> chopsticks, int N, int D)
{
	// first sort the chopsticks in increasing order of length
	// by default the STL sort function sorts in ascending order
	sort(chopsticks.begin(), chopsticks.end());

	// variable to store the answer
	int max_pairs = 0;

	// now start to iterate over the sorted vector
	// and pair the chopsticks
	for(int i = 0; i < N - 1; i++)
	{
		// pair only if the difference in the lengths is
		// at most D
		if((chopsticks[i + 1] - chopsticks[i]) <= D)
		{
			// increase the count
			max_pairs++;

			// increase the value of i
			i++;
		}
	}

	// return the answer
```

```
    return max_pairs;
}

int main()
{
    // take the input
    cout << "Enter the number of chopsticks" << endl;
    int N;
    cin >> N;

    // vector to store the lengths of the chopsticks
    vector <int> chopsticks(N);

    cout << "Enter the lengths of the chopsticks" << endl;

    for(int i = 0; i < N; i++)
    {
        int len;
        cin >> len;

        chopsticks[i] = len;
    }

    // take input the value of D
    cout << "Enter the value of D" << endl;
    int D;
    cin >> D;

    // display the results
    cout << "The maximum number of pairs that can be formed are: " <<
max_chopsticks_pairs(chopsticks, N, D) << endl;
    return 0;
}
```

Output-


Enter the number of chopsticks
5
Enter the lengths of the chopsticks
4 8 5 2 6
Enter the value of D
5
The maximum number of pairs that can be formed are: 2
...Program finished with exit code 0
Press ENTER to exit console.

Experiment no. 21

Title-Implement the tourist guid problem.

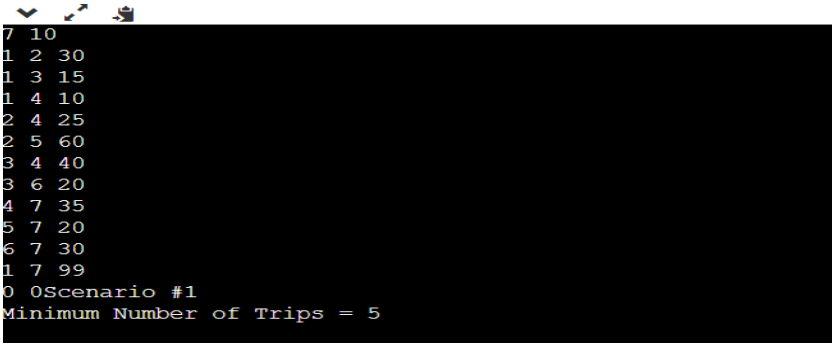
Name :- Rahulkumar Varma Class :- TY CSE

Roll no. :- 3020 Batch :- T1

```
#include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <queue>
#include <stack>
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <cctype>
#include <string>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
using namespace std;

typedef unsigned int uint;
typedef long long int64;
typedef unsigned long long uint64;

#define FOI(i, A, B) for(i=A; i<=B; i++)
#define FOD(i, A, B) for(i=A; i>=B; i--)
```

```
#define PI              acos(-1.0)
#define INF             1<<30
#define EPS             1e-9
#define sqr(x) (x)*(x)
int main(){
        //freopen("testI.txt", "r", stdin);
        //freopen("testO.txt", "w", stdout);
        for (int t = 1; ; t++){
                int N, R;
                scanf("%d%d", &N, &R);
                if (N == 0 && R == 0)
                        break;
                int mat[N][N];
                int i, j, k;
                FOI(i, 0, N-1)
                        FOI(j, 0, N-1)
                                mat[i][j] = 0;
                FOI(i, 1, R){
                        int C1, C2, P;
                        scanf("%d%d%d", &C1, &C2, &P);
                        --C1; --C2;
                        mat[C1][C2] = P;
                        mat[C2][C1] = P;
                }
                FOI(k, 0, N-1)
                        FOI(i, 0, N-1)
                                FOI(j, 0, N-1)
                                        mat[i][j] = mat[j][i] = max(mat[i][j], min(mat[i][k], mat[k][j]));
                int S, D, T;
                scanf("%d%d%d", &S, &D, &T);
                --S; --D;
                printf("Scenario #%d\nMinimum Number of Trips = %d\n\n", t, (int)ceil((double)T /
(mat[S][D] - 1.0)));
        }
        return 0;
}
```

OUTPUT-


7 10
1 2 30
1 3 15
1 4 10
2 4 25
2 5 60
3 4 40
3 6 20
4 7 35
5 7 20
6 7 30
1 7 99
0 0Scenario #1
Minimum Number of Trips = 5

Experiment no. 22

Title-Implement ferry loading problem.

Name :- Rahulkumar Varma Class :- TY CSE

Roll no. :- 3020 Batch :- T1

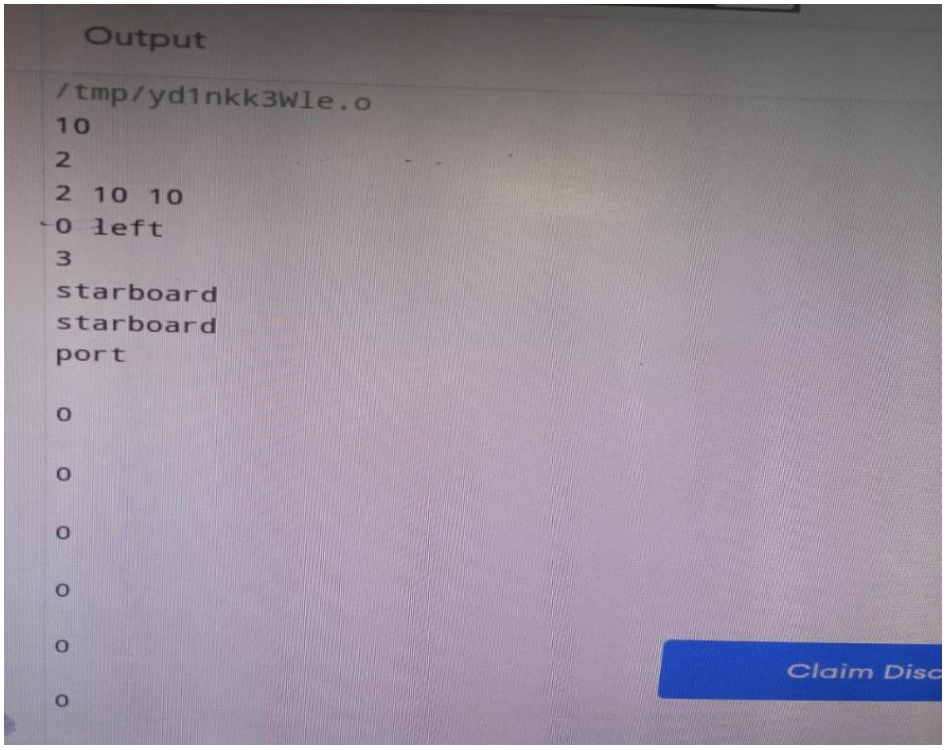
Program:

```
#include <iostream>
#include <string>
#include <vector> using
namespace std;

int main()
{
    int T;  cin
    >> T;
    while ( T-- )
    {
     // Ferry int
     ferry;
     cin >> ferry; ferry *=
     100;

     int car;
     vector<int> cars(1, 0);
     // sum[i] stores the total length of cars 1 to i. vector<int> sum(1,
     0);
     while (cin >> car, car > 0)
     {
          cars.push_back(car); sum.push_back(sum.back() + car);
     }
     // dp[i][j] is true if cars 1 to i could be loaded for length j on right.  vector<vector<bool> >
     dp(cars.size(), vector<bool>(ferry + 1, false)); vector<vector<string> > lane(dp.size(),
     vector<string>(dp[0].size(), "")); pair<int, int> memo = make_pair(0, 0);
     dp[0][0] = true;
       for (int i = 1; i < (int)cars.size(); ++i) for
```

```
(int j = 0; j <= ferry; ++j)
{
    // If cars 1..i-1 were successfully loaded for length j on right.  if (dp[i -
    1][j])
    {
        // Try to load car i on right.  if
        (ferry - j >= cars[i])
```

```
{
                    dp[i][j + cars[i]] = true;
                    lane[i][j + cars[i]] = "starboard" ; memo = make_pair(i, j
                    + cars[i]);
                }
                // Try to load car i on left.
                if (ferry - (sum[i - 1] - j) >= cars[i])
                {
                    dp[i][j] = true;
                    lane[i][j] = "port"; memo = make_pair(i,
                    j);
                }
            }
        }
    cout << memo.first << endl; string print;
    while (lane[memo.first][memo.second] != "")
    {
        print = lane[memo.first][memo.second] + "\n" + print; memo.second -=
        lane[memo.first][memo.second] == "starboard"?
                    cars[memo.first] : 0;
        --memo.first;
    }
    cout << print; if (T)
        cout << endl;
  }
  Return 0;
}
```

Output:-


Output
/tmp/yd1nkk3Wle.o
10
2
2 10 10
0 left
3
starboard
starboard
port
o
o
o
o
o
o
Claim Disc

Experiment No. 23

****

Title: Experiment On Necklace Problem.

Name: Rahulkumar Varma

Class: TE CSE Batch: T1 Roll No: 3020

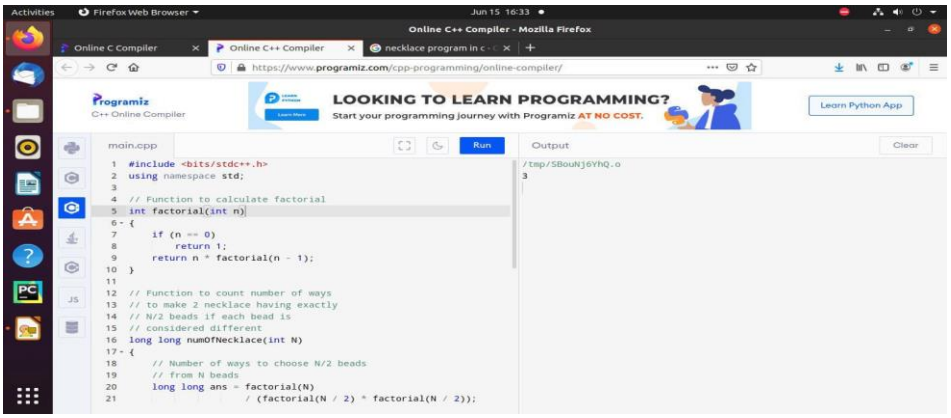
****

Program:

```
#include <bits/stdc++.h>
using namespace std;

// Function to calculate factorial
int factorial(int n)
{
        if (n == 0)
                return 1;
        return n * factorial(n - 1);
}

// Function to count number of ways
// to make 2 necklace having exactly
// N/2 beads if each bead is
// considered different
long long numOfNecklace(int N)
{
        // Number of ways to choose N/2 beads
        // from N beads
        long long ans = factorial(N)
                                / (factorial(N / 2) * factorial(N / 2));

        // Number of ways to permute N/2 beads
        ans = ans * factorial(N / 2 - 1);
        ans = ans * factorial(N / 2 - 1);

        // Divide ans by 2 to remove repetitions
        ans /= 2;

        // Return ans
        return ans;
}
```

```
// Driver Code
int main()
{
        // Given Input
        int N = 4;


// Function Call
                cout <<
                numOfNecklace(N)
                << endl;  return 0;
        }
```

Output:



Experiment no. 24

Title-Implement fire station problem.

Name :- Rahulkumar Varma Class :- TY CSE

Roll no. :- 3020 Batch :- T1

```
#include
<iostream>

                #include <limits>

                int main(void)

                {

                   int firestations, intersections;

                   int i, j, k, saved_value;

                   int cost, min_cost, best_place_to_build_the_goddamn_station;

                   int station, from, to, length; //inputs

                   const int MAX_INTERSECTIONS = 500;

                   //always loop from 1 to the number of intersections

                   int
                distance[MAX_INTERSECTIONS+1][MAX_INTERSECTIONS+1];

                   int distance_to_closest_station[MAX_INTERSECTIONS + 1];

                   const int INFINITY = 100000000;//
                std::numeric_limits<int>::max();



                   //for each test case

                   int test_cases;

                   std::cin >> test_cases;
```

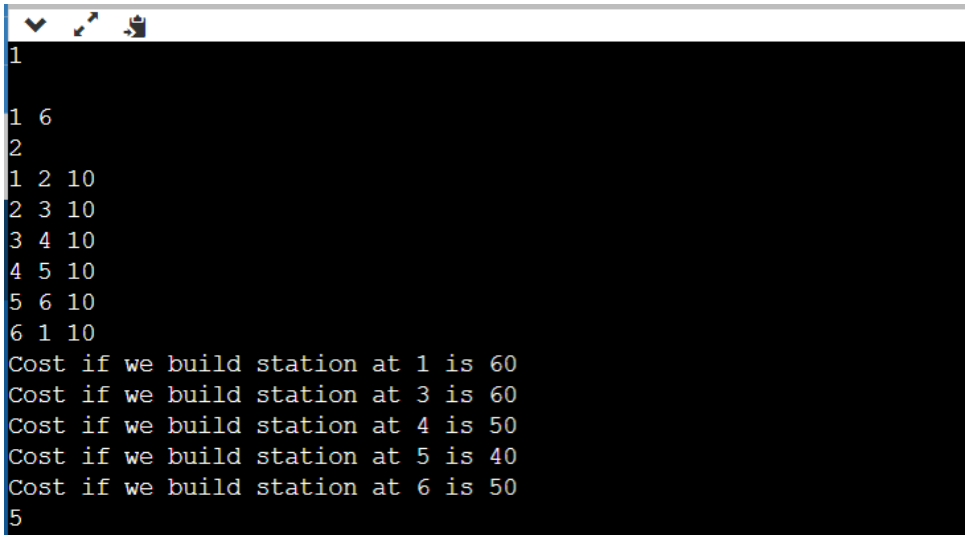
```
  while(test_cases--) {
     std::cin >> firestations >> intersections;



     //set every station's distance to a sentinal value
     for(i = 1; i <= intersections; i++)
        distance_to_closest_station[i] = INFINITY;
     //read stations
     for(i = 1; i <= firestations; i++) {
        std::cin >> station;
        //node is station IIF distance_to_closest_station[node] == 0
        distance_to_closest_station[station] = 0;
     }



     //Floyd-Warshall's algorithm



     //initiate distance between every pair to a sentinal except the path
from a node to itself
     for(i = 1; i <= intersections; i++) {
        for(j = 1; j <= intersections; j++)
           distance[i][j] = INFINITY;
        distance[i][i] = 0;
     }
```

```
//read roads and add edges
for(i = 1; i <= intersections; i++) {
   std::cin >> from >> to >> length;
   distance[from][to] = distance[to][from] = length;
}



//compute shortest paths between every pair
for(k = 1; k <= intersections; k++)
   for(i = 1; i <= intersections; i++)
      for(j = 1; j <= intersections; j++)
         if(distance[i][j] > distance[i][k] + distance[k][j])
            distance[i][j] = distance[i][k] + distance[k][j];
//minimizing cost ("the maximum distance from any intersection
to its nearest fire station")
min_cost = INFINITY;
//for every non-station node, compute cost if it were a station
for(k = 1; k <= intersections; k++) {
   if(distance_to_closest_station[k] == 0) continue;



   //making it a station
   saved_value = distance_to_closest_station[k];
   distance_to_closest_station[k] = 0;
```

```
            for(i = 1; i <= intersections; i++)
                if(distance_to_closest_station[i] != 0)
                    distance_to_closest_station[i] = INFINITY;
            //for every non-station node, update its distance to closest
station
            for(i = 1; i <= intersections; i++)
                for(j = 1; j <= intersections; j++)
                    if(distance_to_closest_station[i] == 0 //if i is a station
                      && distance[i][j] < distance_to_closest_station[j])
                       distance_to_closest_station[j] = distance[i][j];
            //to compute cost
            cost = 0;
            for(i = 1; i <= intersections; i++)
                cost += distance_to_closest_station[i];
            std::cerr << "Cost if we build station at " << k << " is " <<
cost << std::endl;



  if(cost < min_cost) {
                min_cost = cost;
                best_place_to_build_the_goddamn_station = k;
            }
            //restore old distance for next iteration
            distance_to_closest_station[k] = saved_value;
        }
```

```
        std::cout << best_place_to_build_the_goddamn_station <<
    std::endl;

    }

}
```

OUTPUT-



Experiment no. 25

Title-Implement tower of cube problem.

Name :- Rahulkumar Varma Class :- TY CSE

Roll no. :- 3020 Batch :- T1

```
#include <iostream>

using namespace std;

const int Skipped = 6;

int N;

// Is guy we are at and color
// For best face, use 6 to mean next guy (Skipped)
int numberCanStack[505][105], bestTopFaceToUse[505][105];

// Each cube, 0 and 1 pair, 2 and 3 pair, 4 and 5 pair
// so X pairs with  X^1
int cubes[505][6];

string ToFace(int facePos)
{
   switch (facePos)
   {
   case 0:
      return "front";
   case 1:
      return "back";
   case 2:
      return "left";
   case 3:
      return "right";
```

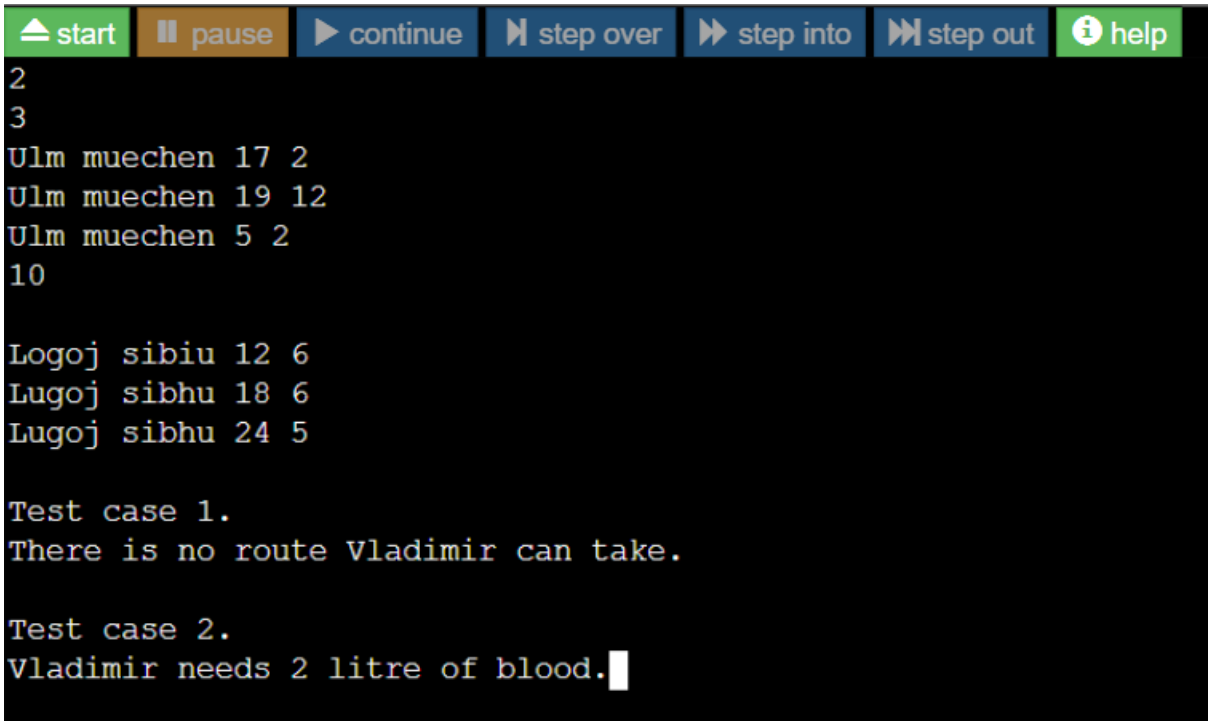
```
    case 4:
        return "top";
    default:
        return "bottom";
    }
}

void PrintOut(int cube, int color)
{
    if (cube == N)
        return;

    if (bestTopFaceToUse[cube][color] == Skipped)
    {
        PrintOut(cube + 1, color);
    }

    else
    {
        int faceUsed = bestTopFaceToUse[cube][color];

        PrintOut(cube + 1, cubes[cube][faceUsed]);

        // Their numbering is weird
        cout << (N - cube) << ' ' << ToFace(faceUsed) << '\n';
    }
}

int NumCanStack(int cube, int color)
{
    if (cube == N)
        return 0;

    int &num = numberCanStack[cube][color];
```

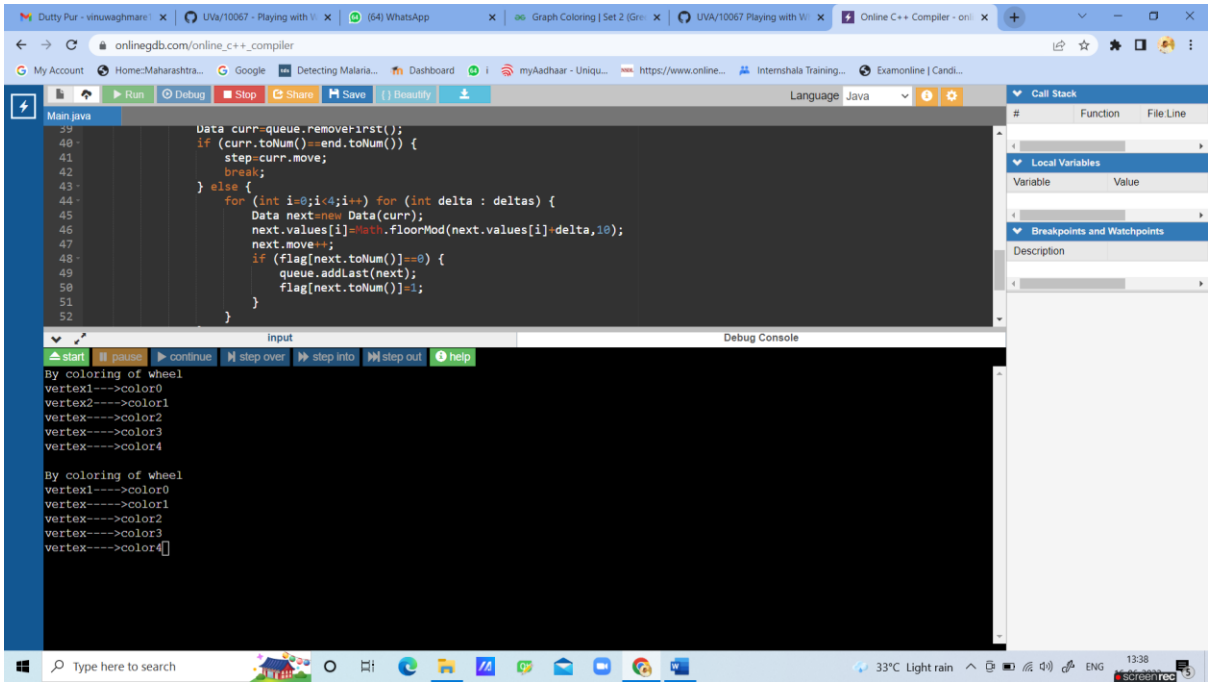
```
    if (num == -1)
    {
        num = 0;
        int &bestTopFace = bestTopFaceToUse[cube][color];
        bestTopFace = Skipped;

        for (int face = 0; face < 6; ++face)
        {
            if (cubes[cube][face] == color)
            {
                int topFace = face^1;
                int amount = NumCanStack(cube + 1, cubes[cube][topFace]) + 1;

                if (amount > num)
                {
                    num = amount;
                    bestTopFace = topFace;
                }
            }
        }

        // Attempt to skip
        int amount = NumCanStack(cube + 1, color);
        if (amount > num)
        {
            num = amount;
            bestTopFace = Skipped;
        }
    }

    return num;
}

int main()
{
```

```
int T = 1;

while (cin >> N, N)
{
   // Read them in reverse
   for (int cube = N - 1; cube >= 0; --cube)
   {
      for (int face = 0; face < 6; ++face)
         cin >> cubes[cube][face];
   }

   // Clear data
   for (int cube = 0; cube < N; ++cube)
   {
      for (int color = 0; color < 105; ++color)
         numberCanStack[cube][color] = -1;
   }

   int bestNum = 0, bestStartCube = -1, bestColor;

   for (int cube = 0; cube < N; ++cube)
   {
      // Have this be the 'bottom' face for this guy
      for (int face = 0; face < 6; ++face)
      {
         int num = NumCanStack(cube, cubes[cube][face]);
         if (num > bestNum)
         {
            bestNum = num;
            bestStartCube = cube;
            bestColor = cubes[cube][face];
         }
      }
   }
```

```
        if (T > 1)
            cout << '\n';
        cout << "Case #" << T++ << '\n';
        cout << bestNum << '\n';

        PrintOut(bestStartCube, bestColor);
    }
}
```

```
Output
/tmp/Ux6azigLz2.o
3
1 2 2 2 1 2
3 3 3 3 3 3
3 2 1 1 1 1
Case #1
2
1 front
3 back
1 0
1 5 1 0 3 6 5
Case #2
1
1 back
2 6 7 3 6 5
5 7 3 2 1 9
1 3 3 5 8 10
6 6 2 2 4 4
1 2 3 4 5 6
10 9 8 7 6 5
Case #3
4
1 back
2 back
3 top
```


start
pause
continue
step over
step into
step out
help
2
3
Ulm muechen 17 2
Ulm muechen 19 12
Ulm muechen 5 2
10
Logoj sibiu 12 6
Lugoj sibhu 18 6
Lugoj sibhu 24 5
Test case 1.
There is no route Vladimir can take.
Test case 2.
Vladimir needs 2 litre of blood.


onlinegdb.com/online_c++_compiler
Run
Debug
Stop
Share
Save
Beautify
Language
Java
Main.java
Data curr=queue.removeFirst();
if (curr.toNum()==end.toNum()) {
step=curr.move;
break;
} else {
for (int i=0;i<4;i++) for (int delta : deltas) {
Data next=new Data(curr);
next.values[i]=Math.floorMod(next.values[i]+delta,10);
next.move++;
if (flag[next.toNum()]==0) {
queue.addLast(next);
flag[next.toNum()]=1;
}
}
Call Stack
Function File:Line
Local Variables
Variable Value
Breakpoints and Watchpoints
Description
input
Debug Console
start
pause
continue
step over
step into
step out
help
By coloring of wheel
vertex1--->color0
vertex2---->color1
vertex---->color2
vertex---->color3
vertex---->color4
By coloring of wheel
vertex1---->color0
vertex----->color1
vertex---->color2
vertex---->color3
vertex---->color4
Type here to search
33°C Light rain
ENG
13:38