**Experiment No. 01**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment of Carmichael Numbers.

**Name:** Mr.Tushar Sarjerao Lohar **Batch:** T1 **Roll No:** 3022

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Program:**

import java.io.\*;

class CFG {

static int gcd(int a, int b)

{

if (a < b)

return gcd(b, a);

if (a % b == 0)

return b;

return gcd(b, a % b);

}

static int power(int x, int y, int mod)

{

if (y == 0)

return 1;

int temp = power(x, y / 2, mod) % mod;

temp = (temp \* temp) % mod;

if (y % 2 == 1)

temp = (temp \* x) % mod;

return temp;

}

static int isCarmichaelNumber(int n)

{

for (int b = 2; b < n; b++) {

if (gcd(b, n) == 1)

if (power(b, n - 1, n) != 1)

return 0;

}

return 1;

}

public static void main(String args[])

{

System.out.println(isCarmichaelNumber(500));

System.out.println(isCarmichaelNumber(561));

System.out.println(isCarmichaelNumber(1105));

}

}

Output:

Text

Description automatically generated

**Experiment No. 02**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment on Smith Number.

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Program**

import java.util.\*;

public class SmithNumberExample1

{

//function finds the sum of digits of its prime factors

static int findSumPrimeFactors(int n)

{

int i=2, sum=0;

while(n>1)

{

if(n%i==0)

{

sum=sum+findSumOfDigit(i);

n=n/i;

}

else

{

do

{

i++;

}

while(!isPrime(i));

}

}

//returns the sum of digits of prime factors

return sum;

}

//function finds the sum of digits of the given numbers

static int findSumOfDigit(int n)

{

//stores the sum

int s=0;

while(n>0)

{

//finds the last digit of the number and add it to the variable s

s=s+n%10;

//removes the last digit from the given number

n=n/10;

}

//returns the sum of digits of the number

return s;

}

//function checks if the factor is prime or not

static boolean isPrime(int k)

{

boolean b=true;

int d=2;

while(d<Math.sqrt(k))

{

if(k%d==0)

{

b=false;

}

d++;

}

return b;

}

//driver code

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter a number: ");

//reads an integer from the user

int n=sc.nextInt();

//calling the userdefined function that finds the sum of digits of the given number

int a = findSumOfDigit(n);

//calling the user-defined function that finds the sum of prime factors

int b = findSumPrimeFactors(n);

System.out.println("Sum of Digits of the given number is = "+a);

System.out.println("Sum of digits of its prime factors is = "+b);

//compare both the sums

if(a==b)

//prints if above condition returns true

System.out.print("The given number is a smith number.");

//prints if above condition returns false

else

System.out.print("The given number is not a smith number.");

}

}

A screenshot of a computer

Description automatically generated with medium confidence

**Experiment No. 03**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment on Euclid Problems.

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Program:**

// Euclidean Algorithm

import java.util.\*;

import java.lang.\*;

class GFG {

// extended Euclidean Algorithm

public static int gcd(int a, int b)

{

if (a == 0)

return b;

return gcd(b % a, a);

}

// Driver Program

public static void main(String[] args)

{

int a = 10, b = 15, g;

g = gcd(a, b);

System.out.println("GCD(" + a + ", " + b + ") = " + g);

a = 35;

b = 10;

g = gcd(a, b);

System.out.println("GCD(" + a + ", " + b + ") = " + g);

a = 31;

b = 2;

g = gcd(a, b);

System.out.println("GCD(" + a + ", " + b + ") = " + g);

}

}

**Output**

Text

Description automatically generated

**Experiment No. 04**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** : Experiment on Light more light.

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Program:**

#include <algorithm>

#include <cstdio>

#include <cmath>

#include <cstring>

#include <deque>

#include <fstream>

#include <iostream>

#include <list>

#include <map>

#include <queue>

#include <set>

#include <stack>

#include <string>

#include <vector>

#include<stdio.h>

using namespace std;

int main()

{

long long int n;

while(scanf("%lld",&n)==1 && n!=0)

{

if (floor(sqrt(n))==ceil(sqrt(n)))

printf("yes\n");

else

printf("no\n");

}

return 0;

}

**Sample Input**

3

6241

8191

0

**Sample Output**

no

yes

no

**Output**:

Text

Description automatically generated

**Experiment No. 05**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment on Tug of War.

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Program:**

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class TugOfWar

{

public int min\_diff;

void TOWUtil(int arr[], int n, boolean curr\_elements[],

int no\_of\_selected\_elements, boolean soln[],

int sum, int curr\_sum, int curr\_position)

{

if (curr\_position == n)

return;

if ((n / 2 - no\_of\_selected\_elements) >

(n - curr\_position))

return;

TOWUtil(arr, n, curr\_elements,

no\_of\_selected\_elements, soln, sum,

curr\_sum, curr\_position+1);

no\_of\_selected\_elements++;

curr\_sum = curr\_sum + arr[curr\_position];

curr\_elements[curr\_position] = true;

if (no\_of\_selected\_elements == n / 2)

{

if (Math.abs(sum / 2 - curr\_sum) <

min\_diff)

{

min\_diff = Math.abs(sum / 2 -

curr\_sum);

for (int i = 0; i < n; i++)

soln[i] = curr\_elements[i];

}

}

else

{

TOWUtil(arr, n, curr\_elements,

no\_of\_selected\_elements,

soln, sum, curr\_sum,

curr\_position + 1);

}

curr\_elements[curr\_position] = false;

}

void tugOfWar(int arr[])

{

int n = arr.length;

boolean[] curr\_elements = new boolean[n];

boolean[] soln = new boolean[n];

min\_diff = Integer.MAX\_VALUE;

int sum = 0;

for (int i = 0; i < n; i++)

{

sum += arr[i];

curr\_elements[i] = soln[i] = false;

}

TOWUtil(arr, n, curr\_elements, 0,

soln, sum, 0, 0);

System.out.print("The first subset is: ");

for (int i = 0; i < n; i++)

{

if (soln[i] == true)

System.out.print(arr[i] + " ");

}

System.out.print("\nThe second subset is: ");

for (int i = 0; i < n; i++)

{

if (soln[i] == false)

System.out.print(arr[i] + " ");

}

}

public static void main (String[] args)

{

int arr[] = {88, 19, 15, -9, 0, 125, 25, -4, 189, 24, -4};

TugOfWar a = new TugOfWar();

a.tugOfWar(arr);

}

}

**Output:**

Text

Description automatically generated

**Experiment No. 06**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment on Factovisor.

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

|  |
| --- |
| #include<stdio.h> |
|  | #include <stdlib.h> |
|  | #include <math.h> |
|  | #include <vector> |
|  | #include <queue> |
|  |  |
|  |  |
|  | using std::vector; |
|  | using std::queue; |
|  |  |
|  |  |
|  | vector<int> primes; |
|  |  |
|  | struct factor { |
|  | int prime, count; |
|  | }; |
|  |  |
|  |  |
|  | /\* Returns true if n is prime. \*/ |
|  | static bool isPrime (int n); |
|  |  |
|  | /\* Tests if n!|m is true. \*/ |
|  | static bool divides\_factorial (const int n, const int m); |
|  |  |
|  | /\* Prepares table of primes. \*/ |
|  | static void prime\_sieve (int max\_n); |
|  |  |
|  | /\* Returns power of factor in n!. \*/ |
|  | static int get\_powers (const unsigned long int n, const int p); |
|  |  |
|  |  |
|  | int main (void) { |
|  | int n, m; |
|  | prime\_sieve(50000); |
|  | while (scanf("%d %d", &n, &m) == 2) { |
|  | if (divides\_factorial(n, m)) { |
|  | printf("%d divides %d!\n", m, n); |
|  | } else { |
|  | printf("%d does not divide %d!\n", m, n); |
|  | } |
|  | } |
|  | return EXIT\_SUCCESS; |
|  | } |
|  |  |
|  | static bool divides\_factorial (const int n, const int m) { |
|  | if (m == 0) { |
|  | return false; |
|  |  |
|  | } else if (n >= m) { |
|  | return true; |
|  |  |
|  | } else { |
|  | int k = m; |
|  | vector<factor> factors; |
|  | for (int i = 0; i < primes.size(); i++) { |
|  | if (primes[i] > k) { |
|  | break; |
|  |  |
|  | } else { |
|  | factor f = {primes[i], 0}; |
|  | while (k % primes[i] == 0) { |
|  | f.count += 1; |
|  | k = k / primes[i]; |
|  | } |
|  |  |
|  | if (f.count) { |
|  | factors.push\_back(f); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | if (k > 1) { // k is a prime |
|  | if (n < k) { |
|  | return false; |
|  | } else { |
|  | factors.push\_back((factor) {k, 1}); |
|  | } |
|  | } |
|  |  |
|  | for (int i = 0; i < factors.size(); i++) { |
|  | if (factors[i].count - get\_powers(n, factors[i].prime) > 0) { |
|  | return false; |
|  | } |
|  | } |
|  | return true; |
|  | } |
|  | } |
|  |  |
|  | static bool isPrime (int n) { |
|  | for (int i = 2; i < n; i++) { |
|  | if (n % i == 0) { |
|  | return false; |
|  | } |
|  | } |
|  | return true; |
|  | } |
|  |  |
|  | static void prime\_sieve (int max\_n) { |
|  | for (int i = 2; i < max\_n; i++) { |
|  | if (isPrime(i)) { |
|  | primes.push\_back(i); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | static int get\_powers (const unsigned long int n, const int p) { |
|  | int res = 0; |
|  | for (unsigned long int power = p; power <= n; power \*= p) { |
|  | res += n / power; |
|  | } |
|  | return res; |
|  | } |

Text

Description automatically generated

**Experiment No. 07**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment on **Summation of Four Primes**

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

|  |
| --- |
| #include<cstdio> |
| #include<math.h> |
| #include<string.h> |
| #define N 10000000 |
| bool P[10000001]; |
| void makePrime(){ |
| memset(P, false, sizeof(P)); |
| int limit = sqrt((double)N); |
| for (int i = 2; i <= limit; i++){ |
| if (!P[i]){ |
| for (int k = (N - 1) / i, j = i\*k; k >= i; k--, j -= i) |
| P[j] = true; |
| } |
| } |
| } |
| void Goldbach(int n){ |
| for (int i = 2;; i++){ |
| if (!P[i] && !P[n - i]){ |
| printf("%d %d", i, n - i); |
| return; |
| } |
| } |
| } |
| int main(){ |
| makePrime(); |
| int n; |
| while (scanf("%d", &n) == 1){ |
| if (n < 8) |
| puts("Impossible."); |
| else{ |
| if (n % 2 == 0) |
| printf("%d %d ", 2, 2), n -= 4; |
| else |
| printf("%d %d ", 2, 3), n -= 5; |
| Goldbach(n); |
| putchar('\n'); |
| } |
| } |
| return 0; |
| } |
| /\* |
|  |
| |  |  | | --- | --- | |  | Sample Input: | |  | 24 | |  | 36 | |  | 46 | |  |  | |  | Sample Output: | |  | 3 11 3 7 | |  | 3 7 13 13 | |  | 11 11 17 7 |   **Output:** |

Text

Description automatically generated

**Experiment No. 08**

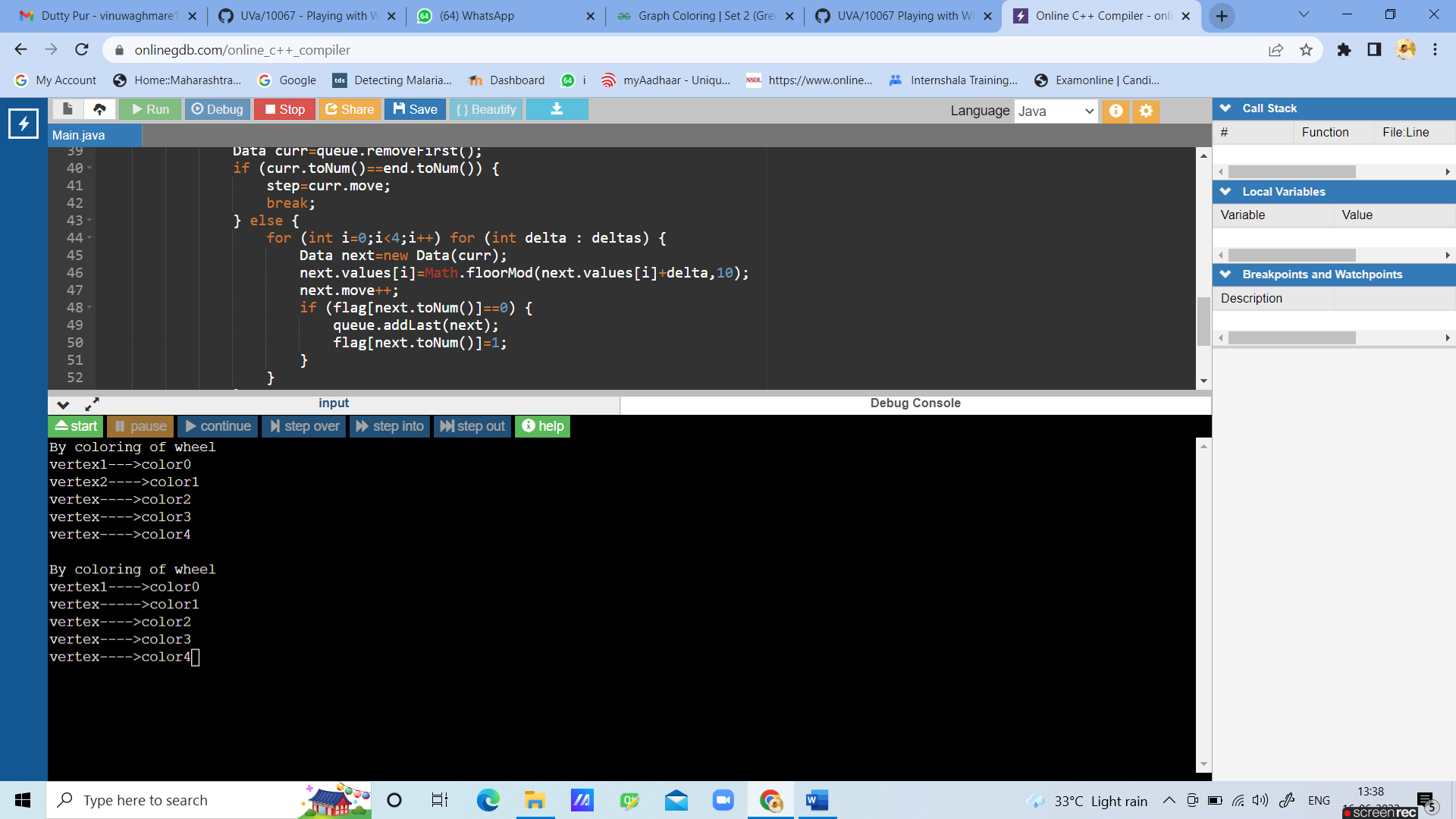
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment on Bicoloring Playing with Wheels.

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

|  |
| --- |
| **Program:**  import java.io.\*; import java.util.\*;  class Main {     public int[][] forbidden;     public int numForbidden;     public int[] start;     public int[] end;         public int bfs(int s1, int s2, int s3, int s4) {         Queue<State> queue = new ArrayDeque<>();         queue.add(new State(s1, s2, s3, s4, 0));                boolean[][][][] visited = new boolean[10][10][10][10];         for (int i = 0; i < numForbidden; i++) {             visited[forbidden[i][0]][forbidden[i][1]][forbidden[i][2]][forbidden[i][3]] = true;         }                while (queue.size() > 0) {             State curr = queue.poll();             int currS1 = curr.s1;             int currS2 = curr.s2;             int currS3 = curr.s3;             int currS4 = curr.s4;             int currNumOp = curr.numOp;                        if (visited[currS1][currS2][currS3][currS4]) {                 continue;             }             visited[currS1][currS2][currS3][currS4] = true;                        if (currS1 == end[0] && currS2 == end[1] && currS3 == end[2] && currS4 == end[3]) {                 return currNumOp;             }                        queue.add(new State((currS1+1)%10, currS2, currS3, currS4, currNumOp+1));             queue.add(new State(currS1, (currS2+1)%10, currS3, currS4, currNumOp+1));             queue.add(new State(currS1, currS2, (currS3+1)%10, currS4, currNumOp+1));             queue.add(new State(currS1, currS2, currS3, (currS4+1)%10, currNumOp+1));             queue.add(new State(((currS1-1)+10)%10, currS2, currS3, currS4, currNumOp+1));             queue.add(new State(currS1, ((currS2-1)+10)%10, currS3, currS4, currNumOp+1));             queue.add(new State(currS1, currS2, ((currS3-1)+10)%10, currS4, currNumOp+1));             queue.add(new State(currS1, currS2, currS3, ((currS4-1)+10)%10, currNumOp+1));         }                return -1;     }        public void process() throws NumberFormatException, IOException {         Scanner sc = new Scanner(System.in);         BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));          int numTests = sc.nextInt();         for (int test = 0; test < numTests; test++) {             start = new int[4];             end = new int[4];             for (int i = 0; i < 4; i++) {                 start[i] = sc.nextInt();             }             for (int i = 0; i < 4; i++) {                 end[i] = sc.nextInt();             }                        numForbidden = sc.nextInt();             forbidden = new int[numForbidden][4];             for (int i = 0; i < numForbidden; i++) {                 for (int j = 0; j < 4; j++) {                     forbidden[i][j] = sc.nextInt();                 }             }                        bw.write(bfs(start[0], start[1], start[2], start[3])+"\n");         }                                                       bw.flush();         bw.close();                return;     }        public static void main(String[] args) throws NumberFormatException, IOException {         Main m = new Main();         m.process();                System.exit(0);     } }  class State {     int s1;     int s2;     int s3;     int s4;     int numOp;        public State(int s1, int s2, int s3, int s4, int numOp) {         this.s1 = s1;         this.s2 = s2;         this.s3 = s3;         this.s4 = s4;         this.numOp = numOp;     } } |



**Experiment No. 09**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Experiment on From Dusk Till Dawn.

**Name:** Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

**Program:**

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

new Main().start();

}

private void start() {

Scanner in = new Scanner(System.in);

int total, m;

String[] start\_city = new String[1000];

String[] arrive\_city = new String[1000];

int[] start\_time = new int[1000];

int[] arrive\_time = new int[1000];

int cost\_time;

int [][]map;

String from, to;

total = in.nextInt();

for (int cas = 1; cas <= total; cas++) {

System.out.println("Test Case " + cas + ".");

m = in.nextInt();

for (int i = 0; i < m; i++) {

start\_city[i] = in.next();

arrive\_city[i] = in.next();

start\_time[i] = in.nextInt() % 24;

cost\_time = in.nextInt();

if ((cost\_time > 12 || (cost\_time == 12 && start\_time[i] != 18)) || (start\_time[i] > 6 && start\_time[i] < 18)) {

i--;

m--;

continue;

}

arrive\_time[i] = (start\_time[i] + cost\_time) % 24;

if (arrive\_time[i] > 6 && arrive\_time[i] < 18) {

i--;

m--;

continue;

}

}

map = new int[m + 2][m + 2];

from = in.next();

to = in.next();

if (from.equals(to)) {

System.out.println("Vladimir needs 0 litre(s) of blood.");

continue;

}

for (int i = 0; i < map.length; i++) {

Arrays.fill(map[i], -1);

}

for (int i = 0; i < m ; i++) {

if (start\_city[i].equals(from)) {

map[0][i + 1] = 0;

}

if (arrive\_city[i].equals(to)) {

map[i + 1][m + 1] = 0;

}

for (int j = 0; j < m; j++) {

if (i == j || !arrive\_city[i].equals(start\_city[j])) {

continue;

}

if (f(arrive\_time[i]) <= f(start\_time[j])) {

map[i + 1][j + 1] = 0;

} else {

map[i + 1][j + 1] = 1;

}

}

}

int cost = dij(map);

if (cost == Integer.MAX\_VALUE) {

System.out.println("There is no route Vladimir can take.");

} else {

System.out.println("Vladimir needs " + cost + " litre(s) of blood.");

}

}

}

private int dij(int[][] map) {

int n = map.length;

int best[] = new int[n];

boolean[] used = new boolean[n];

Arrays.fill(best, Integer.MAX\_VALUE);

best[0] = 0;

for (int i = 0; i < n && !used[n - 1]; i++) {

int min = Integer.MAX\_VALUE;

int id = -1;

for (int j = 0; j < n; j++) {

if (!used[j] && best[j] < min ) {

id = j;

min = best[j];

}

}

if (id == -1) {

break;

}

used[id] = true;

for (int j = 0; j < n; j++) {

if (map[id][j] != -1) {

best[j] = Math.min(best[j], best[id] + map[id][j]);

}

}

}

return best[n - 1];

}

private int f(int time) {

if (time <= 6) {

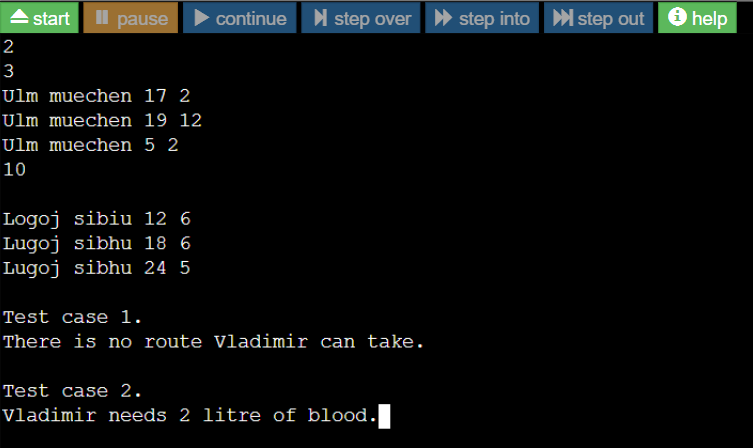
time += 24;

}

return time;

}

}



**Experiment No. 10**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:**Experiment on coins

**Name: :**Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

**Program:**

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

new Main().start();

}

private void start() {

Scanner in = new Scanner(System.in);

int total, m;

String[] start\_city = new String[1000];

String[] arrive\_city = new String[1000];

int[] start\_time = new int[1000];

int[] arrive\_time = new int[1000];

int cost\_time;

int [][]map;

String from, to;

total = in.nextInt();

for (int cas = 1; cas <= total; cas++) {

System.out.println("Test Case " + cas + ".");

m = in.nextInt();

for (int i = 0; i < m; i++) {

start\_city[i] = in.next();

arrive\_city[i] = in.next();

start\_time[i] = in.nextInt() % 24;

cost\_time = in.nextInt();

if ((cost\_time > 12 || (cost\_time == 12 && start\_time[i] != 18)) || (start\_time[i] > 6 && start\_time[i] < 18)) {

i--;

m--;

continue;

}

arrive\_time[i] = (start\_time[i] + cost\_time) % 24;

if (arrive\_time[i] > 6 && arrive\_time[i] < 18) {

i--;

m--;

continue;

}

}

map = new int[m + 2][m + 2];

from = in.next();

to = in.next();

if (from.equals(to)) {

System.out.println("Vladimir needs 0 litre(s) of blood.");

continue;

}

for (int i = 0; i < map.length; i++) {

Arrays.fill(map[i], -1);

}

for (int i = 0; i < m ; i++) {

if (start\_city[i].equals(from)) {

map[0][i + 1] = 0;

}

if (arrive\_city[i].equals(to)) {

map[i + 1][m + 1] = 0;

}

for (int j = 0; j < m; j++) {

if (i == j || !arrive\_city[i].equals(start\_city[j])) {

continue;

}

if (f(arrive\_time[i]) <= f(start\_time[j])) {

map[i + 1][j + 1] = 0;

} else {

map[i + 1][j + 1] = 1;

}

}

}

int cost = dij(map);

if (cost == Integer.MAX\_VALUE) {

System.out.println("There is no route Vladimir can take.");

} else {

System.out.println("Vladimir needs " + cost + " litre(s) of blood.");

}

}

}

private int dij(int[][] map) {

int n = map.length;

int best[] = new int[n];

boolean[] used = new boolean[n];

Arrays.fill(best, Integer.MAX\_VALUE);

best[0] = 0;

for (int i = 0; i < n && !used[n - 1]; i++) {

int min = Integer.MAX\_VALUE;

int id = -1;

for (int j = 0; j < n; j++) {

if (!used[j] && best[j] < min ) {

id = j;

min = best[j];

}

}

if (id == -1) {

break;

}

used[id] = true;

for (int j = 0; j < n; j++) {

if (map[id][j] != -1) {

best[j] = Math.min(best[j], best[id] + map[id][j]);

}

}

}

return best[n - 1];

}

private int f(int time) {

if (time <= 6) {

time += 24;

}

return time;

}

}

#include<bits/stdc++.h>

using namespace std;

// m is size of coins array (number of different coins)

int minCoins(int coins[], int m, int V)

{

// base case

if (V == 0) return 0;

// Initialize result

int res = INT\_MAX;

// Try every coin that has smaller value than V

for (int i=0; i<m; i++)

{

if (coins[i] <= V)

{

int sub\_res = minCoins(coins, m, V-coins[i]);

// Check for INT\_MAX to avoid overflow and see if

// result can minimized

if (sub\_res != INT\_MAX && sub\_res + 1 < res)

res = sub\_res + 1;

}

}

return res;

}

// Driver program to test above function

int main()

{

int coins[] = {9, 6, 5, 1};

int m = sizeof(coins)/sizeof(coins[0]);

int V = 11;

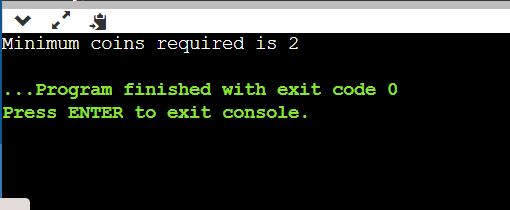
cout << "Minimum coins required is "

<< minCoins(coins, m, V);

return 0;

}

**Output-**

****

**Experiment No. 11**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:** Minimum spanning tree algorithm

**Name: :**Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

#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 },

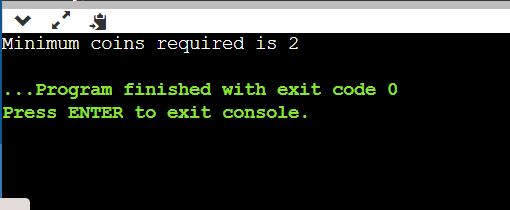
                        { 0, 0, 5, 0, 0 } };

    prims\_MST(graph);

    return 0;

}

**OUTPUT**



**Experiment No. 12**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:**Finding shortest path using Dijkstra’s algorithm

**Name: :**Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

#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 },

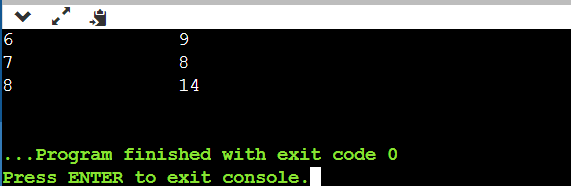
{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

dijkstra(graph, 0);

return 0;

}

OUTPUT-



**Experiment No. 13**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:**Experiment on jolly jumper sequencing.

**Name: :**Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

// 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};

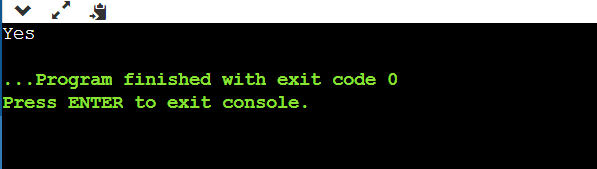
int n = sizeof(a)/ sizeof(a[0]);

isJolly(a, n)? cout << "Yes" : cout << "No";

return 0;

}

OUTPUT-



**Experiment No. 14**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title:**Experiment on tower Hanoi.

**Name: :**Mr.Tushar Sarjerao Lohar

**Class:** TE CSE **Batch:** T1 **Roll No:** 3022

#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()

{

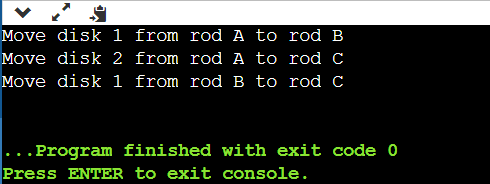
int n = 4; // Number of disks

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 :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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

}

}

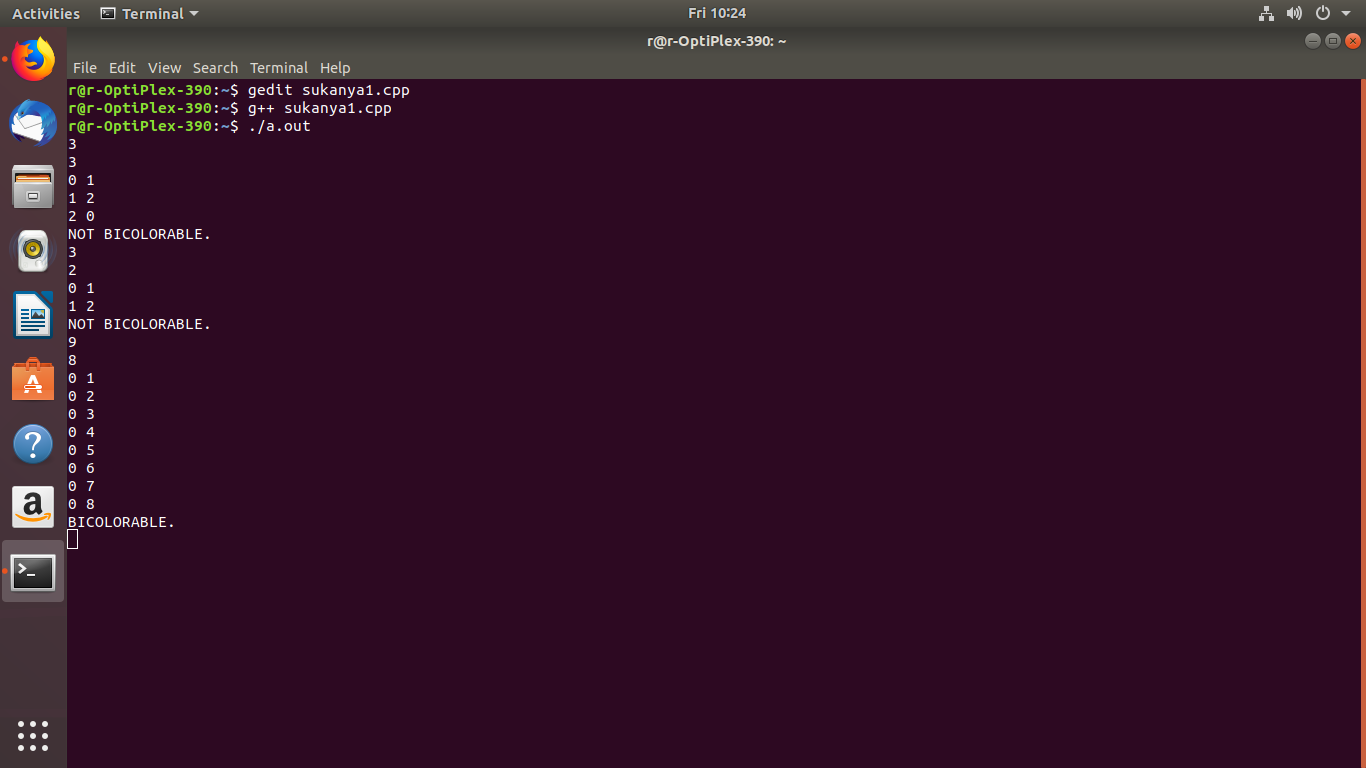
if(check == NO)

printf("BICOLORABLE.\n");

}

return 0;

}

OUTPUT :-

**Experiment no. 16**

Title-Expiriment on problem is Bigger and smarter.

Name :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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;

}

}

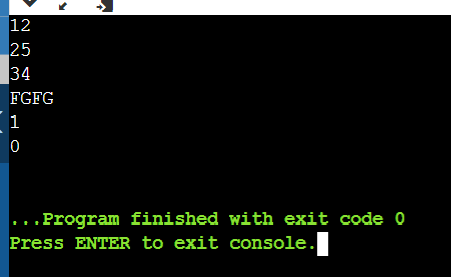
printf("%d\n",ans);

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 :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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;

}

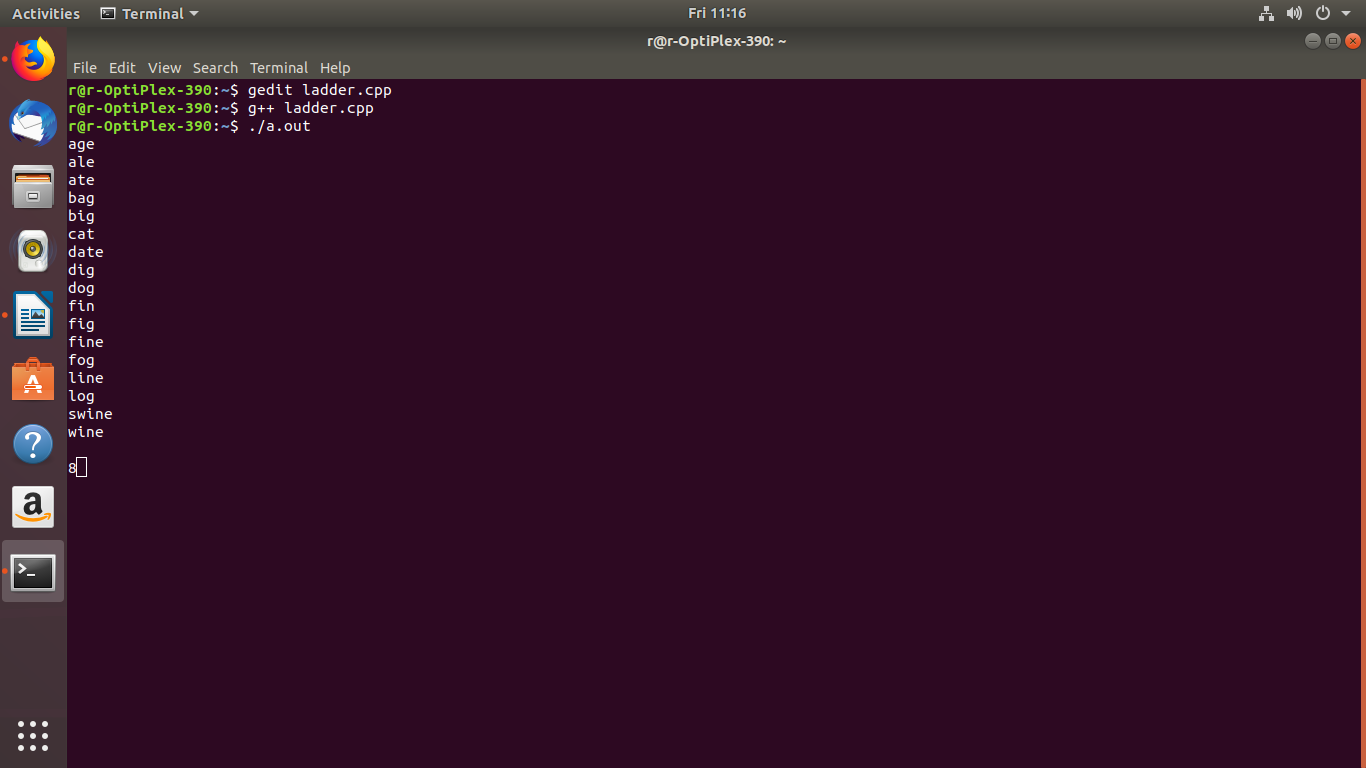
}

cout << wordcounts<< endl;

return 0;

}

OUTPUT :



**Experiment no. 18**

Title-Implement distinct subsequent problem.

Name :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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

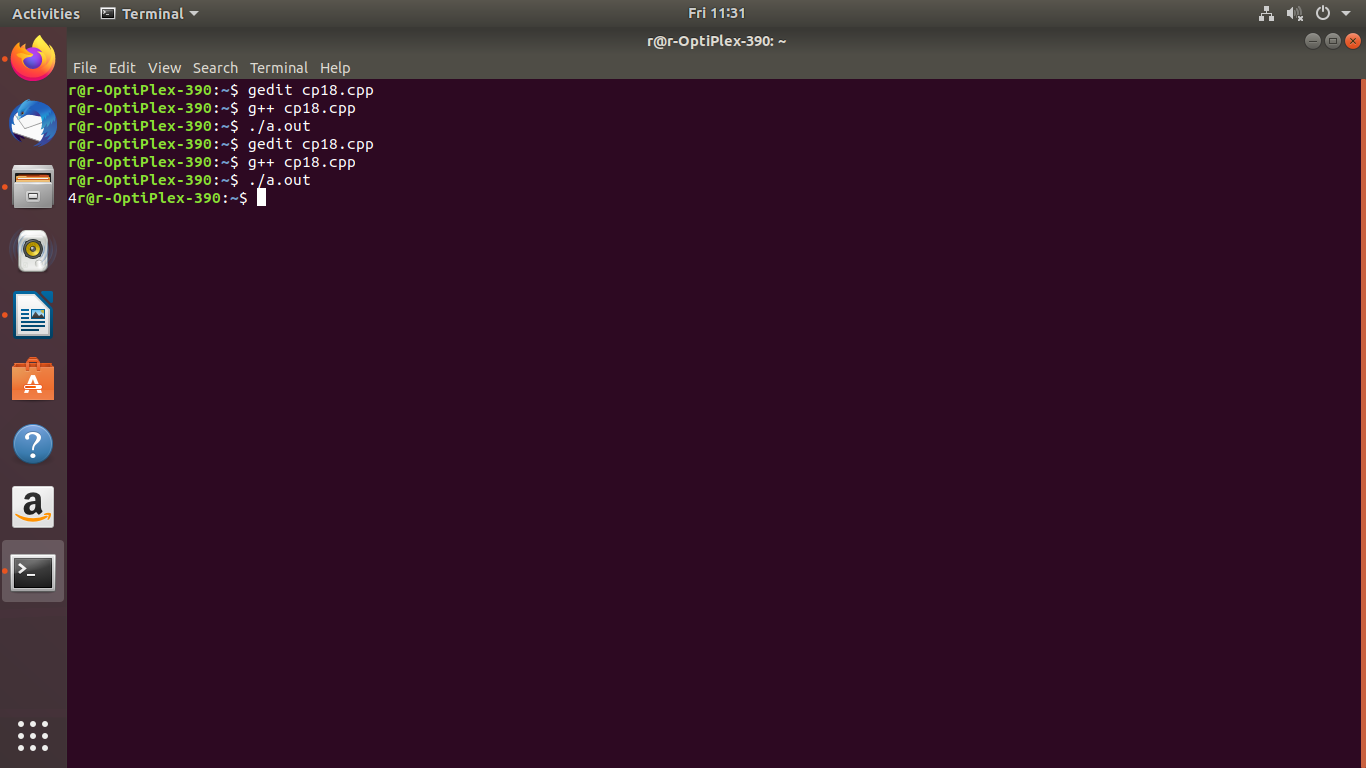
cout << sn.size();

sn.clear();

return 0;

// This code is contributed by Kishan Mishra

}

OUTPUT :-

**Experiment no. 19**

Title-Implement weight and measure problem.

Name :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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;

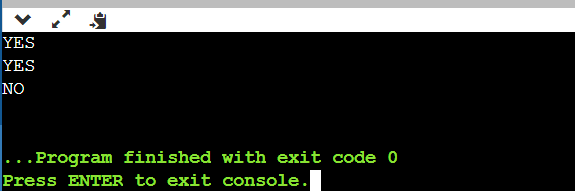
else

cout << "NO" << endl;

return 0;

}

Output-

-

**Experiment no. 20**

Title-Implement chopstick problem.

Name :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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;

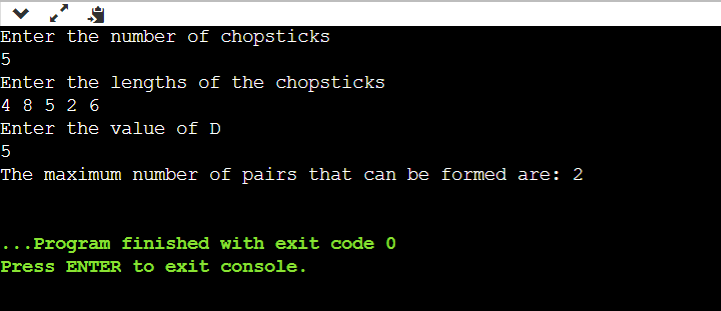
        // display the results

        cout << "The maximum number of pairs that can be formed are: " << max\_chopsticks\_pairs(chopsticks, N, D) << endl;

**return** 0;

}

Output-



**Experiment no. 21**

Title-Implement the tourist guid problem.

Name :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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;

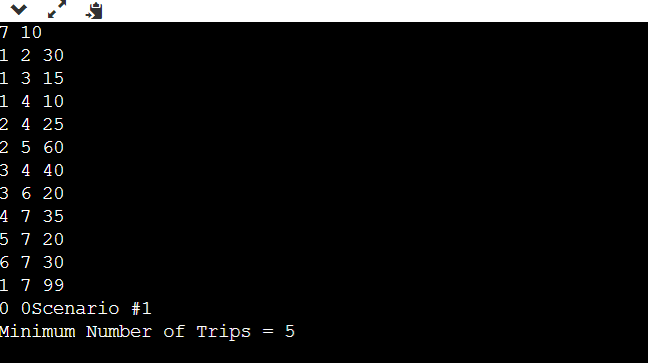
printf("Scenario #%d\nMinimum Number of Trips = %d\n\n", t, (int)ceil((double)T / (mat[S][D] - 1.0)));

}

return 0;

}

**OUTPUT-**

****

**Experiment no. 22**

Title-Implement ferry loading problem.

Name :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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)

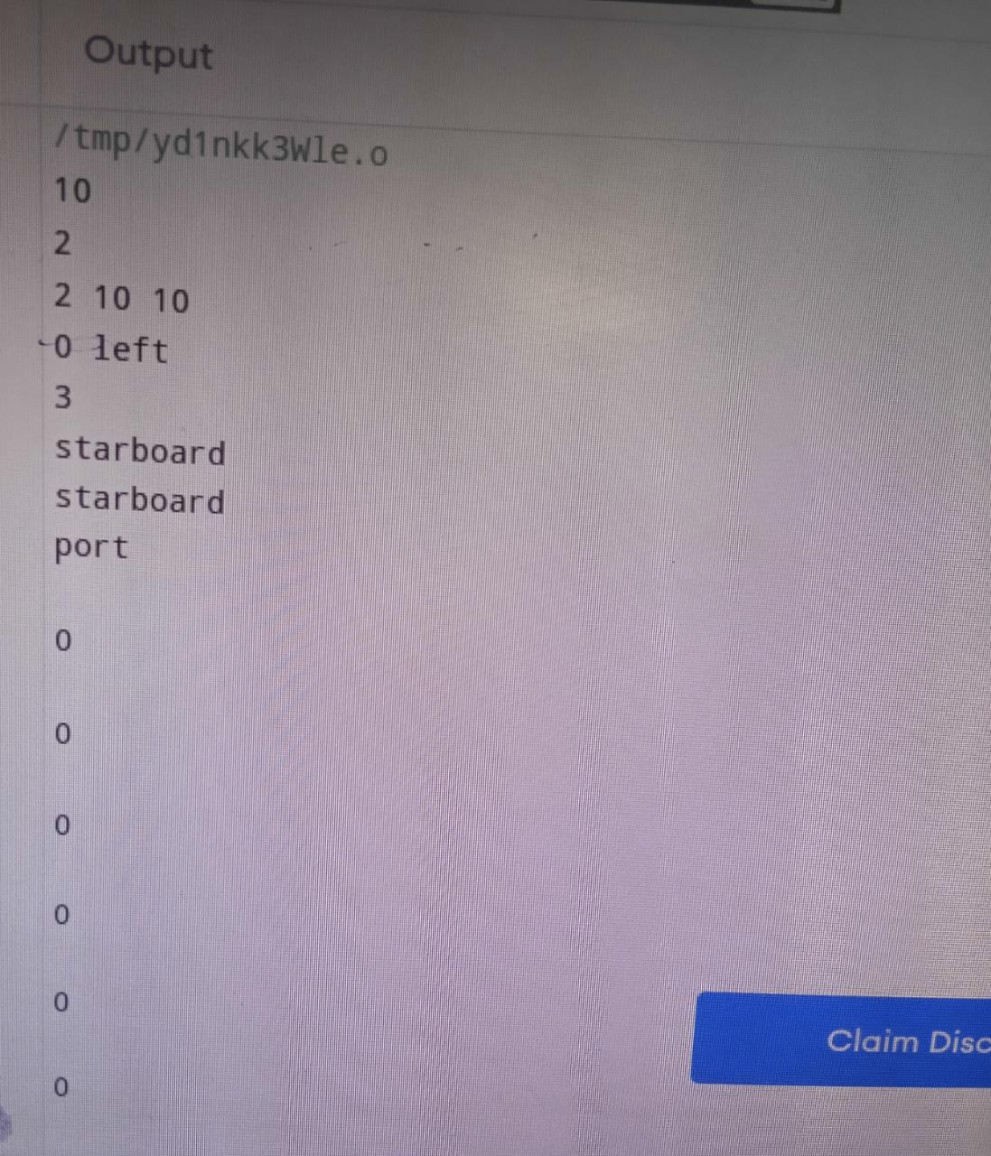
cout << endl;

}

Return 0;

}

Output:-



Experiment No. 23

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Title: Experiment On Necklace Problem.

Name: Mr.Tushar Sarjerao Lohar

Class: TE CSE Batch: T1 Roll No: 3022

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 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()

{

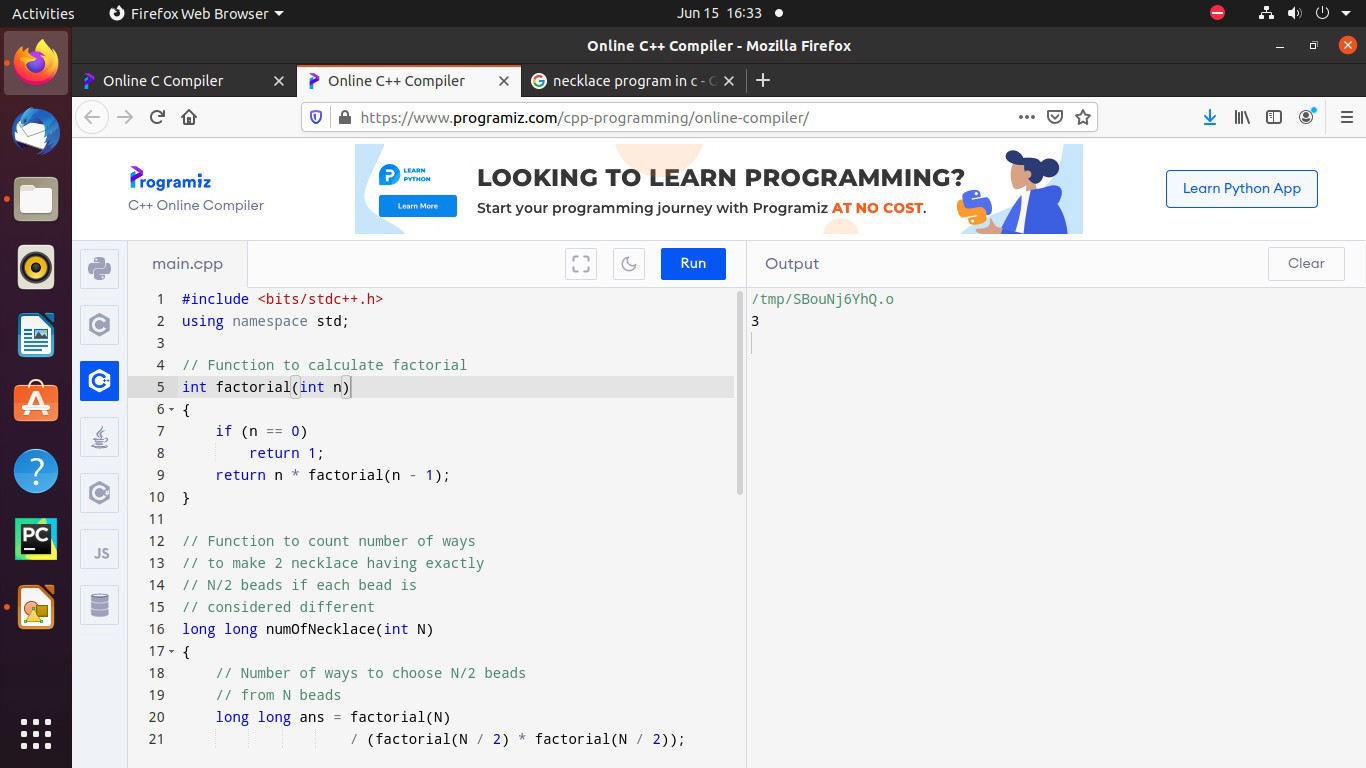
// Given Input int N = 4;

// Function Call

cout << numOfNecklace(N) << endl; return 0;

}

# Output:



**Experiment no. 24**

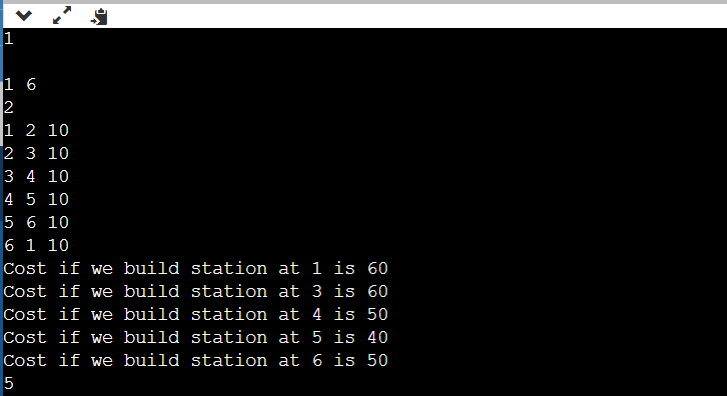
Title-Implement fire station problem.

Name :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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 :- Mr.Tushar Sarjerao Lohar Class :- TY CSE

Roll no. :- 3022 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);

}

}

