Experiment No. 01

****

Title: Experiment of Camichael Numbers.

Name: Rahulkumar Varma

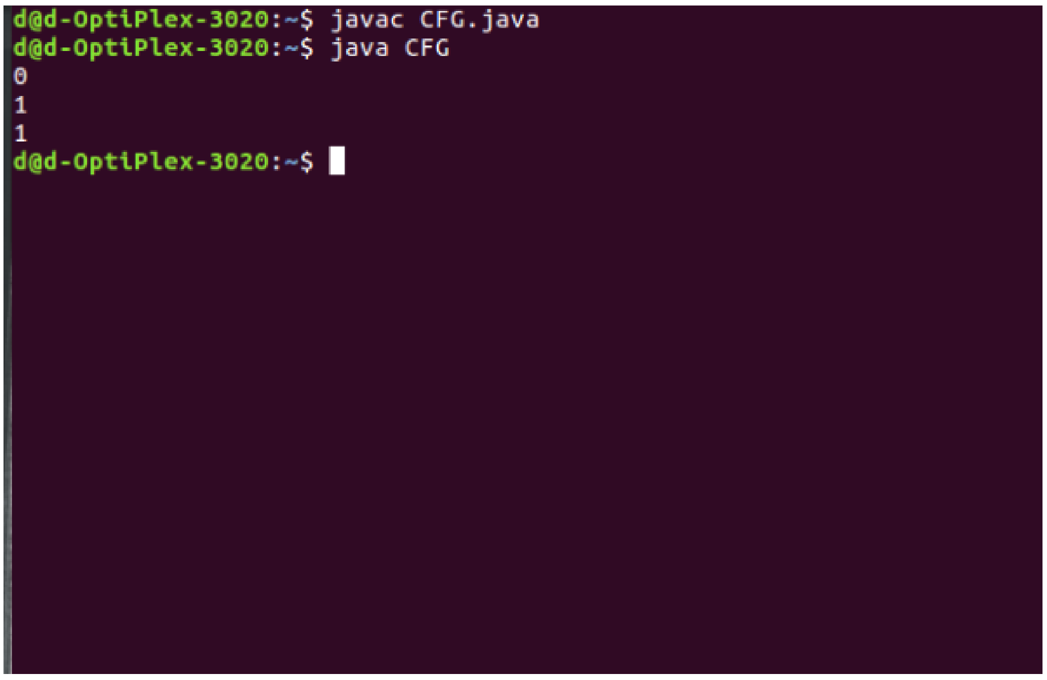
Class: TE CSE **Batch:** T1 **Roll No:** 3020

****

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:



```
d@d-OptiPlex-3020:~$ javac CFG.java
d@d-OptiPlex-3020:~$ java CFG
0
1
1
d@d-OptiPlex-3020:~$
```

Experiment No. 02

****

Title: Experiment on Smith Number.

Name: Rahulkumar Varma

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

****

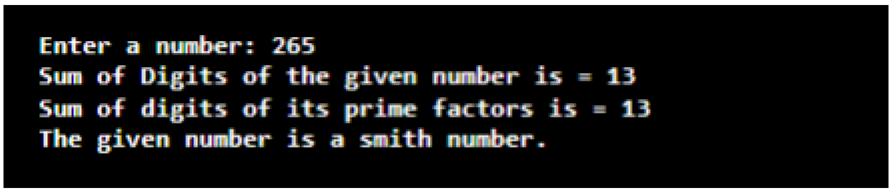
Program

```
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[])
```

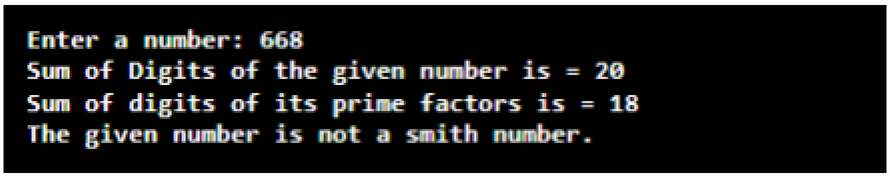
```
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter a number: ");
//reads an integer from the user
int n=sc.nextInt();
//calling the userdefined function that finds the sum of digits of the given number
int a = findSumOfDigit(n);
//calling the user-defined function that finds the sum of 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.");
}
}
```

Output 1:



```
Enter a number: 265
Sum of Digits of the given number is = 13
Sum of digits of its prime factors is = 13
The given number is a smith number.
```

Output 2:



```
Enter a number: 668
Sum of Digits of the given number is = 20
Sum of digits of its prime factors is = 18
The given number is not a smith number.
```

Experiment No. 03

****

Title: Experiment on Euclid Problems.

Name: Rahulkumar Varma

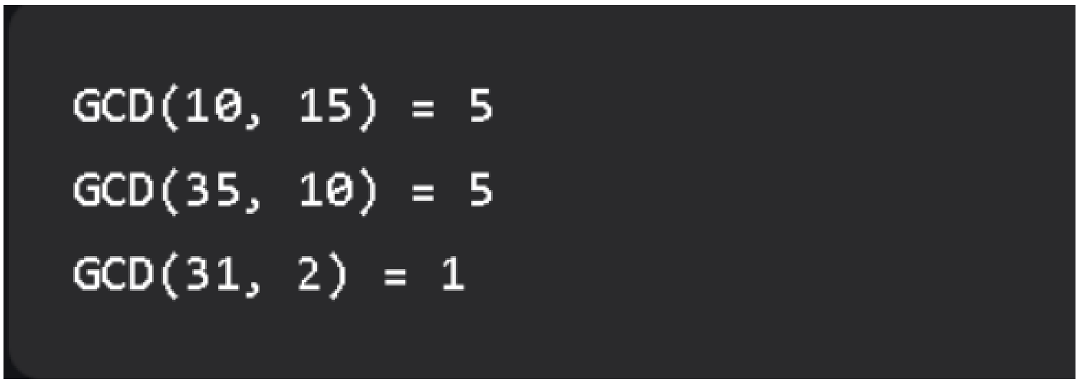
Class: TE CSE **Batch:** T1 **Roll No:** 3020

****

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



```
GCD(10, 15) = 5
GCD(35, 10) = 5
GCD(31, 2) = 1
```

Experiment No. 04

****

Title: : Experiment on Light more light.

Name: Rahulkumar Varma
Class: TE CSE **Batch:** T1 **Roll No:** 3020

****

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

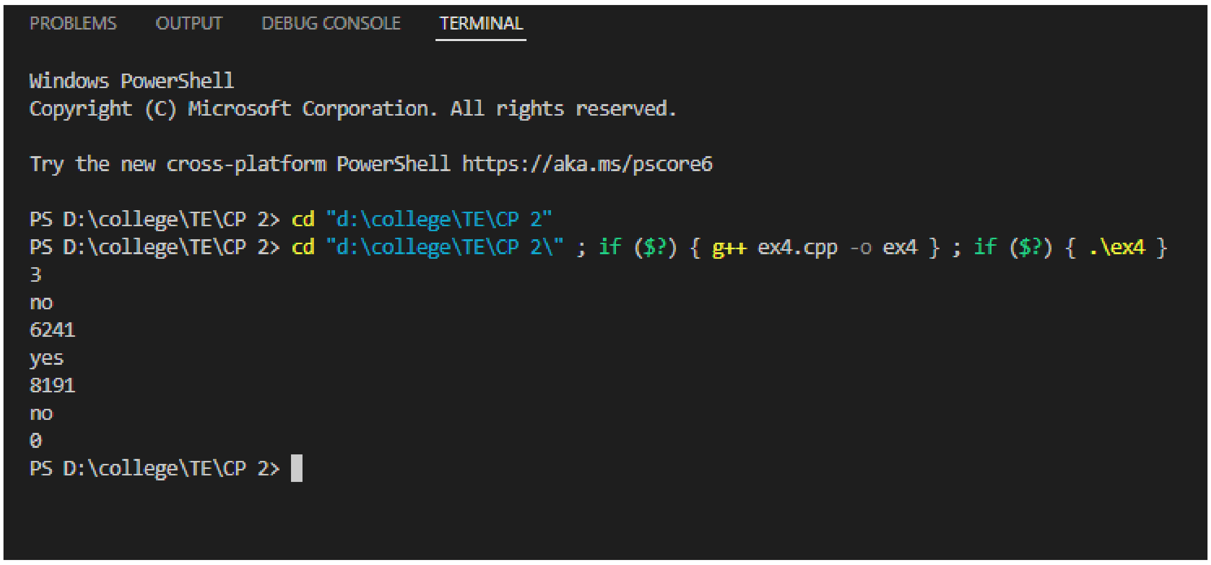
Sample Output

no

yes

no

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\college\TE\CP 2> cd "d:\college\TE\CP 2"
PS D:\college\TE\CP 2> cd "d:\college\TE\CP 2\" ; if ($?) { g++ ex4.cpp -o ex4 } ; if ($?) { .\ex4 }
3
no
6241
yes
8191
no
0
PS D:\college\TE\CP 2> 
```

Experiment No. 05

****

Title: Experiment on Tug of War.

Name: Rahulkumar Varma

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

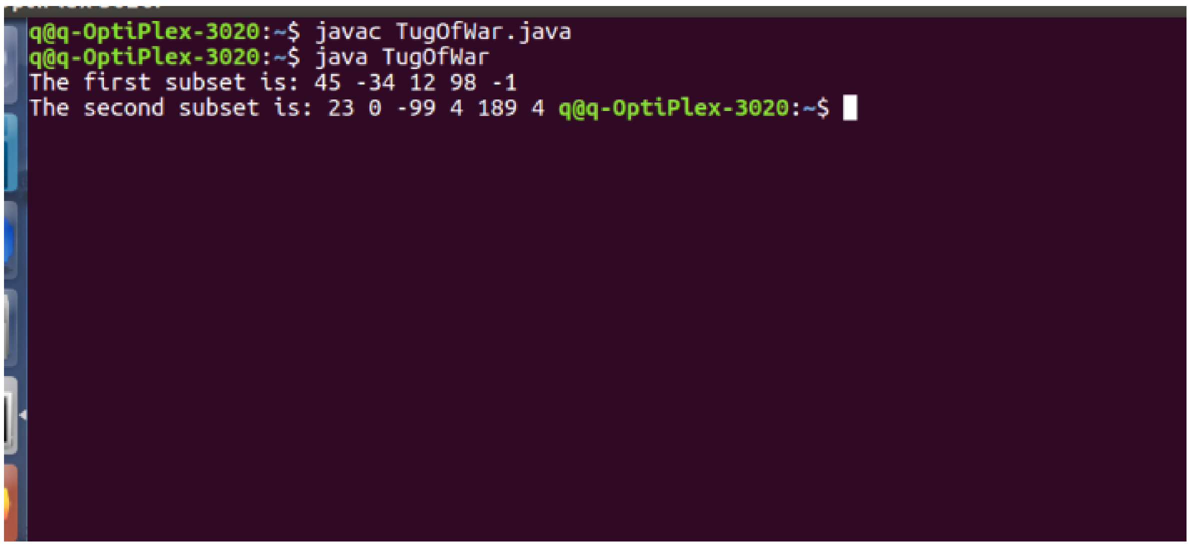
****

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:



```
q@q-OptiPlex-3020:~$ javac TugOfWar.java
q@q-OptiPlex-3020:~$ java TugOfWar
The first subset is: 45 -34 12 98 -1
The second subset is: 23 0 -99 4 189 4 q@q-OptiPlex-3020:~$
```

Experiment No. 06

****

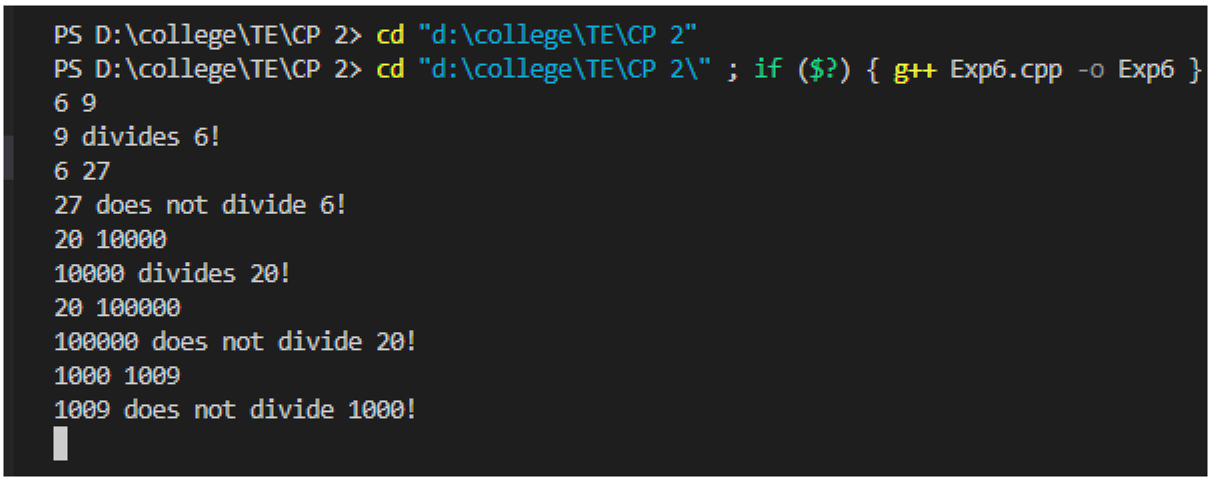
Title: Experiment on Factovisor.

Name: Rahulkumar Varma

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

```
#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;
}
```



```
PS D:\college\TE\CP 2> cd "d:\college\TE\CP 2"
PS D:\college\TE\CP 2> cd "d:\college\TE\CP 2\" ; if ($?) { g++ Exp6.cpp -o Exp6 }
6 9
9 divides 6!
6 27
27 does not divide 6!
20 10000
10000 divides 20!
20 100000
100000 does not divide 20!
1000 1009
1009 does not divide 1000!
```

Experiment No. 07

****

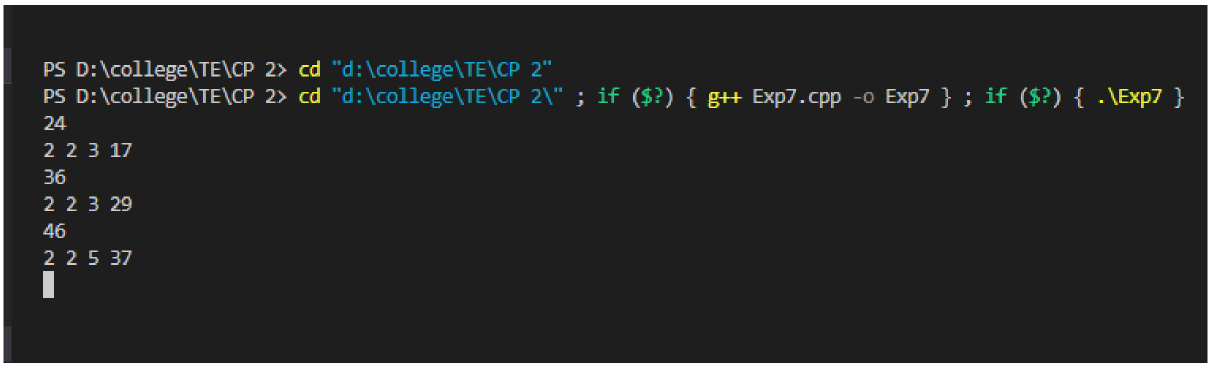
Title: Experiment on **Summation of Four Primes**

Name: Rahulkumar Varma

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

```
#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:



```
PS D:\college\TE\CP 2> cd "d:\college\TE\CP 2"
PS D:\college\TE\CP 2> cd "d:\college\TE\CP 2\" ; if ($?) { g++ Exp7.cpp -o Exp7 } ; if ($?) { .\Exp7 }
24
2 2 3 17
36
2 2 3 29
46
2 2 5 37
```

Experiment No. 08

****

Title: Experiment on Bicoloring Playing with Wheels.

Name: Rahulkumar Varma

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

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);
    }
}
```

```java
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: Rahulkumar Varma

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

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;
```

```java
            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;
    }
}
```