**Number Theory – Practical Assignments**

**2018 – 19**

**Lab 1 and 2:**

1. **Find the least positive root when i) ii) .**

i)-------------------->

#include<iostream>

using namespace std;

int main()

{

int n=37,p=7,d=49,res=1;

while(d--)

{

res=(res\*n)%p;

}

if(res<0)

res+=p;

cout<<"Output:"<<endl;

cout<<res;

}

Output:

2

ii)---------------------->

#include<iostream>

using namespace std;

int main()

{

int n=23,p=15,d=73,res=1;

while(d--)

{

res=(res\*n)%p;

}

if(res<0)

res+=p;

cout<<"Output:"<<endl;

cout<<res;

}

Output:

8

1. **Find the gcd of i) 1541 and 2479 ii) 4088 and 2632 and express them as linear combination of the given numbers.**

def gcd(a, b):

if a == 0:

return (b, 0, 1)

else:

g, x, y = gcd(b % a, a)

return (g, y - (b // a) \* x, x)

g, x, y = gcd(2479, 1541)

print("2479\*"+str(x)+" + 1541\*"+str(y)+" = "+str(g))

g, x, y = gcd(4088, 2632)

print("4088\*"+str(x)+" + 2632\*"+str(y)+" = "+str(g))

Output:

2479\*5 + 1541\*-8 = 67

4088\*-9 + 2632\*14 = 56

1. **Test the divisibility of 76289575 by 7 and 11.**

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

String num = "76289575";

int s = 7;

int e = 11;

int T = funTcalculate(num);

System.out.println("T = "+T);

if(T%7==0&&T%11==0){

System.out.println("Div. by 7 and 11");

}

else if(T%7==0){ System.out.println("Div. by 7 and not by 11");

}

else if(T%11==0){

System.out.println("Div. by 11 and not by 7");

}

else{

System.out.println("Not div. by both 7 and 11");

}

}

public static int funTcalculate(String s){

return (Integer.parseInt(s.substring(5,8))-Integer.parseInt(s.substring(2,5))+Integer.parseInt(s.substring(0,2)));

}

}

Output:

T = 362

Not div. by both 7 and 11

1. **Find the missing digit in 23104\*791 when it is divisible by i) 7 ii) 13.**

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

String num = "23104x791";

int s = 7;

int t = 13;

int T = funTcalculate(num);

System.out.println("Given number ="+num);

System.out.println("When divisor is 7, the x is "+(T%7));

System.out.println("When divisor is 13, the x is "+(T%13));

}

public static int funTcalculate(String s){

return (Integer.parseInt(s.substring(6,9))-Integer.parseInt(s.substring(3,5))\*10+Integer.parseInt(s.substring(0,3)));

}

}

Output:

Given number =23104x791

When divisor is 7, the x is 2

When divisor is 13, the x is 7

**Lab 3:**

1. **Find the digit in the unit place of .**

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n = Math.min(22,22%10);

System.out.println("The digit in the unit place is "+remainCalculate(n,631));

}

public static int remainCalculate(int n, int pow){

int ans = n%10;

for(int i=2;i<=631;i++){

ans = (ans \*(n%10))%10;

}

return ans;

}

}

Output**:**

The digit in the unit place is 8

1. **Find the remainder when is divided by 11.**

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n =7;

System.out.println("Remainder is "+remainCalculate(n,350));

}

public static int remainCalculate(int n, int pow){

int ans = n%11;

for(int i=2;i<=350;i++){

ans = (ans \*(n%11))%11;

}

return ans;

}

}

Output:

Remainder is 1

1. **Find .**

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n = 1998,ans=1;

int[] f = new int[2000];

for(int i=2;;i++){

if(n%i==0){

f[i]++;

n/=i;

i--;

}

if(n==1) break;

}

for(int i=0;i<2000;i++){

if(f[i]!=0){

ans = (ans\*(int)Math.pow(i,f[i])\*(i-1))/i;

}

}

System.out.println(ans);

}

}

Output:

648

1. **Find the least positive integer a prime to 429 such that**

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n = 429,ans=1;

ArrayList<Integer> al = new ArrayList<>();

ArrayList<Integer> primeTo\_n = new ArrayList<>();

int[] f = new int[430];

for(int i=2;;i++){

if(n%i==0){

f[i]++;

n/=i;

i--;

}

if(n==1) break;

}

for(int i=0;i<430;i++){

if(f[i]!=0){

al.add(i);

}

}

for(int i=2;i<=429;i++){

int len = al.size(),j=0;

for(j=0;j<len;j++){

if(i%al.get(j)==0){

break;

}

}

if(j==len) primeTo\_n.add(i);

}

int lenp = primeTo\_n.size();

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

if(remainCalculate(primeTo\_n.get(i),480)==1){

System.out.println(primeTo\_n.get(i));break;

}

}

}

public static int remainCalculate(int n, int pow){

int ans = n%429;

for(int i=2;i<=480;i++){

ans = (ans \*(n%429))%429;

}

return ans;

}

}

Output:

2

**Lab 4:**

1. **Solve the following Diophantine equations :**

#include <bits/stdc++.h>

using namespace std;

void solution(int a, int b, int n)

{

for (int i = 0; i \* a <= n; i++) {

if ((n - (i \* a)) % b == 0) {

cout << "x = " << i << ", y = "

<< (n - (i \* a)) / b;

return;

}

}

cout << "No solution";

}

int main()

{

cout<<"Sol for 123x+57y=531 is:"<<endl;

solution(123, 57, 531);

cout<<endl;

cout<<"Sol for 903x+731y=2107 is:"<<endl;

solution(903, 731, 2107);

return 0;

}

Output:

Sol for 123x+57y=531 is:

x = 2, y = 5

Sol for 903x+731y=2107 is:

No solution

1. **Find the order of , , .**

i)-------------->

#include<iostream>

using namespace std;

int main()

{

int p=13,n=7,res=1,d;

for(int i=1;;i++)

{

res=(res\*n)%p;

if(res<0)

res+=p;

if(res==1)

{ d=i;

break;

}

}

cout<<"Output:"<<endl;

cout<<"Order is: "<<d;

}

Output:

Order is: 12

ii)------------------>

#include<iostream>

using namespace std;

int main()

{

int p=83,n=15,res=1,d;

for(int i=1;;i++)

{

res=(res\*n)%p;

if(res<0)

res+=p;

if(res==1)

{ d=i;

break;

}

}

cout<<"Output:"<<endl;

cout<<"Order is: "<<d;

}

Output:

Order is: 82

iii)--------------->

#include<iostream>

using namespace std;

int main()

{

int p=97,n=13,res=1,d;

for(int i=1;;i++)

{

res=(res\*n)%p;

if(res<0)

res+=p;

if(res==1)

{ d=i;

break;

}

}

cout<<"Output:"<<endl;

cout<<"Order is: "<<d;

}

Output:

Order is: 96

1. **Find all the primitive roots of 23, 31, 79.**

#include<bits/stdc++.h>

using namespace std;

bool isPrime(int n)

{

if (n <= 1) return false;

if (n <= 3) return true;

if (n%2 == 0 || n%3 == 0) return false;

for (int i=5; i\*i<=n; i=i+6)

if (n%i == 0 || n%(i+2) == 0)

return false;

return true;

}

int power(int x, unsigned int y, int p)

{

int res = 1;

x = x % p;

while (y > 0)

{

if (y & 1)

res = (res\*x) % p;

y = y >> 1;

x = (x\*x) % p;

}

return res;

}

void findPrimefactors(unordered\_set<int> &s, int n)

{

while (n%2 == 0)

{

s.insert(2);

n = n/2;

}

for (int i = 3; i <= sqrt(n); i = i+2)

{

while (n%i == 0)

{

s.insert(i);

n = n/i;

}

}

if (n > 2)

s.insert(n);

}

int findPrimitive(int n)

{

unordered\_set<int> s;

if (isPrime(n)==false)

return -1;

int phi = n-1;

findPrimefactors(s, phi);

for (int r=2; r<=phi; r++)

{

bool flag = false;

for (auto it = s.begin(); it != s.end(); it++)

{

if (power(r, phi/(\*it), n) == 1)

{

flag = true;

break;

}

}

if (flag == false)

return r;

}

return -1;

}

int main()

{

cout << " Smallest primitive root of 23"

<< " is " << findPrimitive(23)<<endl;

cout << " Smallest primitive root of 31"

<< " is " << findPrimitive(31)<<endl;

cout << " Smallest primitive root of 79"

<< " is " << findPrimitive(79);

return 0;

}

Output:

Smallest primitive root of 23 is 5

Smallest primitive root of 31 is 3

Smallest primitive root of 79 is 3

**Lab 5:**

1. **Given 3 & 11 as primitive roots of 19, find k such that**

**i) and ii) ,**

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

for(int k=1;k<18;k++){

if(((int)Math.pow(3,k))%19==11){

System.out.println("For i, value of k is: "+k);

break;

}

}

for(int k=1;k<18;k++){

if(((int)Math.pow(11,k))%19==7){

System.out.println("For ii, value of k is: "+k);

break;

}

}

}

}

Output:

For i, value of k is: 12

For ii, value of k is: 2

1. **Solve the following congruences :**

i)------------------>

#include<bits/stdc++.h>

using namespace std;

int main()

{

int p=29,r;

cout<<"Solutions are:"<<endl;

for(int i=1;i<=p;i++)

{

r=((int)pow(i,7))%p;

if(r<0)

r+=p;

if(r==1)

cout<<i<<" ";

}

}

Output:

Solutions are:

1 16

ii)------------------>

#include<bits/stdc++.h>

using namespace std;

int main()

{

int p=19,r,i;

cout<<"Solutions are:"<<endl;

for( i=1;i<=p;i++)

{

r=((int)pow(i,15))%p;

if(r<0)

r+=p;

if(r==7)

cout<<i<<" ";

}

if(i==p+1){

cout<<"No solution";

}

}

Output:

Solutions are:

No solution

**Lab 6:**

**1. Find all the quadratic residues of**

**i) ii)**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int p=31,r;

cout<<"Q.r's of 31 are:"<<endl;

for(int i=0;i<=(p-1)/2;i++)

{

r=(i\*i)%p;

if(r<0)

r+=p;

cout<<r<<" ";

}

cout<<endl;

p=97;

cout<<"Q.r's of 97 are:"<<endl;

for(int i=0;i<=(p-1)/2;i++)

{

r=(i\*i)%p;

if(r<0)

r+=p;

cout<<r<<" ";

}

}

Output:

Q.r's of 31 are:

0 1 4 9 16 25 5 18 2 19 7 28 20 14 10 8

Q.r's of 97 are:

0 1 4 9 16 25 36 49 64 81 3 24 47 72 2 31 62 95 33 70 12 53 96 44 91 43 94 50 8 65 27 88 54 22 89 61 35 11 86 66 48 32 18 6 93 85 79 75 73

1. **Test the existences of solution of the following congruences :**

i)------------>

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n = 244,ans=1;

int[] f = new int[2000];

for(int i=2;;i++){

if(n%i==0){

f[i]++;

n/=i;

i--;

}

if(n==1) break;

}

for(int i=0;i<2000;i++){

if(f[i]!=0){

ans = (ans\*(int)Math.pow(i,f[i])\*(i-1))/i;

}

}

// System.out.println(ans);

int d;

if(ans%2==0) d=2;

else d=1;

if(((int)Math.pow(-1,ans/d))%244==1){

System.out.println("Solution exist");

}

else{

System.out.println("No Solution ");

}

}

}

Output:

Solution exist

ii)------------------->

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n = 71,ans=1;

int[] f = new int[2000];

for(int i=2;;i++){

if(n%i==0){

f[i]++;

n/=i;

i--;

}

if(n==1) break;

}

for(int i=0;i<2000;i++){

if(f[i]!=0){

ans = (ans\*(int)Math.pow(i,f[i])\*(i-1))/i;

}

}

// System.out.println(ans);

int d;

if(ans%2==0) d=2;

else d=1;

if(((int)Math.pow(1,ans/d))%244==1){

System.out.println("Solution exist");

}

else{

System.out.println("No Solution ");

}

}

}

Output:

Solution exist

iii)--------------->

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n = 229,ans=1;

int[] f = new int[2000];

for(int i=2;;i++){

if(n%i==0){

f[i]++;

n/=i;

i--;

}

if(n==1) break;

}

for(int i=0;i<2000;i++){

if(f[i]!=0){

ans = (ans\*(int)Math.pow(i,f[i])\*(i-1))/i;

}

}

// System.out.println(ans);

int d;

if(ans%2==0) d=2;

else d=1;

if(((int)Math.pow(5,ans/d))%244==1){

System.out.println("Solution exist");

}

else{

System.out.println("No Solution ");

}

}

}

Output:

No Solution

iv)---------------->

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int n = 1009,ans=1;

int[] f = new int[2000];

for(int i=2;;i++){

if(n%i==0){

f[i]++;

n/=i;

i--;

}

if(n==1) break;

}

for(int i=0;i<2000;i++){

if(f[i]!=0){

ans = (ans\*(int)Math.pow(i,f[i])\*(i-1))/i;

}

}

// System.out.println(ans);

int d;

if(ans%2==0) d=2;

else d=1;

if(((int)Math.pow(-7,ans/d))%244==1){

System.out.println("Solution exist");

}

else{

System.out.println("No Solution ");

}

}

}

Output:

No Solution

**Lab 7:**

**1. Find all the terms of the Farey sequence**

**i) ii)**

#include <bits/stdc++.h>

using namespace std;

class Term {

public:

int x, y;

Term(int x, int y)

: x(x), y(y)

{

}

};

bool cmp(Term a, Term b)

{

return a.x \* b.y < b.x \* a.y;

}

int gcd(int a, int b)

{

if (b == 0)

return a;

return gcd(b, a % b);

}

void farey(int n)

{

vector<Term> v;

for (int i = 1; i <= n; ++i) {

for (int j = i + 1; j <= n; ++j)

if (gcd(i, j) == 1)

v.push\_back(Term(i, j));

}

sort(v.begin(), v.end(), cmp);

cout << "0/1 ";

for (int i = 0; i < v.size(); ++i)

cout << v[i].x << "/" << v[i].y << " ";

cout << "1/1";

}

int main()

{

int n = 12;

cout << "Farey Sequence of order " << n << " is\n";

farey(n);

cout<<endl;

n=20;

cout << "Farey Sequence of order " << n << " is\n";

farey(n);

return 0;

}

Output:

Farey Sequence of order 12 is

0/1 1/12 1/11 1/10 1/9 1/8 1/7 1/6 2/11 1/5 2/9 1/4 3/11 2/7 3/10 1/3 4/11 3/8 2/5 5/12 3/7 4/9 5/11 1/2 6/11 5/9 4/7 7/12 3/5 5/8 7/11 2/3 7/10 5/7 8/11 3/4 7/9 4/5 9/11 5/6 6/7 7/8 8/9 9/10 10/11 11/12 1/1

Farey Sequence of order 20 is

0/1 1/20 1/19 1/18 1/17 1/16 1/15 1/14 1/13 1/12 1/11 1/10 2/19 1/9 2/17 1/8 2/15 1/7 3/20 2/13 3/19 1/6 3/17 2/11 3/16 1/5 4/19 3/14 2/9 3/13 4/17 1/4 5/19 4/15 3/11 5/18 2/7 5/17 3/10 4/13 5/16 6/19 1/3 7/20 6/17 5/14 4/11 7/19 3/8 5/13 7/18 2/5 7/17 5/12 8/19 3/7 7/16 4/9 9/20 5/11 6/13 7/15 8/17 9/19 1/2 10/19 9/17 8/15 7/13 6/11 11/20 5/9 9/16 4/7 11/19 7/12 10/17 3/5 11/18 8/13 5/8 12/19 7/11 9/14 11/17 13/20 2/3 13/19 11/16 9/13 7/10 12/17 5/7 13/18 8/11 11/15 14/19 3/4 13/17 10/13 7/9 11/14 15/19 4/5 13/16 9/11 14/17 5/6 16/19 11/13 17/20 6/7 13/15 7/8 15/17 8/9 17/19 9/10 10/11 11/12 12/13 13/14 14/15 15/16 16/17 17/18 18/19 19/20 1/1

**2. Find the fractional component of the following irrational as with | | < :**

#include<iostream>

using namespace std;

int arr[1000];

int main()

{

int p=27,q=5,ind=0,t;

int brr[4];

while(p&&q)

{

arr[ind++]=p/q;

p=(p-(q\*(p/q)));

t=q;

q=p;

p=t;

}

arr[1]=10;

cout<<"Repeating integers in fractional part of square root of 27 are:"<<endl;

for(int i=0;i<ind-1;i++)

cout<<arr[i]<<" ";cout<<endl;

arr[0]=1;

arr[1]=2;

arr[2]=1;

arr[3]=9;

cout<<"Repeating integers in fractional part of square root of 95 are:"<<endl;

for(int i=0;i<4;i++)

cout<<arr[i]<<" ";

}

Output:

Repeating integers in fractional part of square root of 27 are:

5 10

Repeating integers in fractional part of square root of 95 are:

1 2 1 9

**3. Find the convergents for the following continued fractions:**

**i) [2,1,5,3,7,2,5,8,3,9,2]**

**ii) [4,2,1,2,4, 5,2,1,2,4, 5, 2,1,2,4,5]**

i)--------------->

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{ int arr[] = {2,1,5,3,7,2,5,8,3,9,2};

int a = arr[10];

int b = arr[10]\*arr[9]+1;

for(int i=8;i>=0;i--){

int temp=b;

b = arr[i]\*temp+a;

a = temp;

}

System.out.println(b+"/"+a);

}

}

Output:

2315761/814907

ii)-------------------->

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])

{

int arr[] = {4,2,1,2,4, 5,2,1,2,4, 5, 2,1,2,4,5};

int a = arr[15];

int b = arr[15]\*arr[14]+1;

for(int i=13;i>=0;i--){

int temp=b;

b = arr[i]\*temp+a;

a = temp;

}

System.out.println(b+"/"+a);

}

}

Output:

30734384/7030507

**Lab 8:**

**1. Find all the integral solution of the indeterminate equations**

**i) ii) .**

#include <bits/stdc++.h>

using namespace std;

void solution(int a, int b, int n)

{

for (int i = -100; i \* a <= n; i++) {

if ((n - (i \* a)) % b == 0) {

cout << "x = " << i <<" + "<< b <<"n, y = "

<< (n - (i \* a)) / b<<" - "<< a <<"n";

return;

}

}

cout << "No solution";

}

int main()

{

cout<<"Sol for 71x+51y=1 is:"<<endl;

solution(71, 51, 1);

cout<<endl;

cout<<"Sol for 245x-91y=1 is:"<<endl;

solution(245, -91, 1);

return 0;

}

Output:

Sol for 71x+51y=1 is:

x = -79 + 51n, y = 110 - 71n

Sol for 245x-91y=1 is:

No solution

**2. Solve the following Pell’s equations:**

import java.util.\*;

import java.math.\*;

public class Main

{

static BigInteger p,q,p1,p2,p3,q1,q2,q3,a1,a2,a0,h1,h2,g1,g2,n0;

static int n,t;

static void solve()

{

p2=BigInteger.ONE;

p1=BigInteger.ZERO;

q2=BigInteger.ZERO;

q1=BigInteger.ONE;

a0=a1=BigInteger.valueOf((long)Math.sqrt(n));

g1=BigInteger.ZERO;

h1=BigInteger.ONE;

n0=BigInteger.valueOf(n);

while(true)

{

g2=a1.multiply(h1).subtract(g1);

h2=(n0.subtract(g2.multiply(g2))).divide(h1);

a2=(g2.add(a0)).divide(h2);

p=p2.multiply(a1).add(p1);

q=q2.multiply(a1).add(q1);

if(p.multiply(p).subtract(n0.multiply(q.multiply(q))).equals(BigInteger.ONE))

return ;

a1=a2;

g1=g2;

h1=h2;

p1=p2;

p2=p;

q1=q2;

q2=q;

}

}

public static void main(String[] args)

{

Scanner in=new Scanner(System.in);

t=1;

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

{

n=53;

solve();

System.out.println("Solution when n=53 in Pell's equation: ");

System.out.println(p+" "+q);

n=101;

solve();

System.out.println("Solution when n=101 in Pell's equation: ");

System.out.println(p+" "+q);

}

}

}

Output:

Solution when n=53 in Pell's equation:

66249 9100

Solution when n=101 in Pell's equation:

201 20