**Assignment – 3**

**Q1. Write a program to find the list of generators present in Zn\*where n is a large integer.**

**A:**

*Pseudocode and Explanation –*

func bin() – // function used to convert an integer to binary digits

func easy\_mod() - // function used to calculate *ae mod n*  for large integers.

* Function calls bin to convert an integer to binary form. Then apply repeated square and multiply algorithm for exponentiation to calculate *ae mod n* .

func gcd() - // function to find the gcd of two numbers

func main() –

* We find Z­n­\* and calulate phi(n) i.e. the number of elements of Z­n­\* .
* Next, we calculate all the factors of phi(n).
* Next, the outer loop traverses through all the elements in Z­n­\*. The inner loop traverses through all the elements in factors. Now, I calculated order of each element using a data structure ***map*.** This allowed me to store the different type of order that can occur and their corresponding Z­n­\* value.
* Lastly I just printed the map value for phi.

*Code -*

#include<bits/stdc++.h>

using *namespace* std;

*void* bin(*unsigned* *n*, vector<*int*> *&vec*){

    if (*n* > 1)

        bin(*n* / 2, *vec*);

*int* x = *n* % 2;

*vec*.push\_back(x);

}

*int* easy\_mod(*int* *a*, *int* *e*, *int* *n*){

    vector<*int*> bin\_repr;

    bin(*e*,bin\_repr);

    reverse(bin\_repr.begin(),bin\_repr.end());

    // for(int i=0;i<bin\_repr.size();i++){

    //     cout<<bin\_repr[i]<<" ";

    // }

*int* A = *a*;

*int* b;

    if(bin\_repr[0] == 1){

        b = A;

    }else{

        b = 1;

    }

    for(*int* i=1;i<bin\_repr.size();i++){

        A = (A\*A)%*n*;

        if(bin\_repr[i] == 1){

            b = (A\*b)%*n*;

        }

    }

    // cout<<b;

    return b;

}

*int* gcd(*int* *a*, *int* *b*)

{

    // Find Minimum of a and b

*int* result = min(*a*, *b*);

    while (result > 0) {

        if (*a* % result == 0 && *b* % result == 0) {

            break;

        }

        result--;

    }

    // Return gcd of a and b

    return result;

}

*int* main()

{

*int* n;

    cout<<"Enter a number: "<<endl;

    cin>>n;

    vector<*int*> zn\_star;

    for(*int* i=1;i<=n-1;i++){

        if(gcd(i,n) == 1){

            zn\_star.push\_back(i);

        }

    }

*int* phi = zn\_star.size();

    // cout<<phi;

    vector<*int*> factors;

    for(*int* i=1;i<=phi;i++){

        if(phi%i == 0){

            factors.push\_back(i);

            // cout<<i<<" ";

        }

    }

    unordered\_map<*int*, vector<*int*>> order;

    for(*int* i=0;i<zn\_star.size();i++){

        for(*int* j=0;j<factors.size();j++){

*int* x = zn\_star[i];

*int* y = factors[j];

*int* val = easy\_mod(x,y,n);

            if(val == 1){

                order[y].push\_back(x);

                // cout<< x <<" "<< y<<endl;

                break;

            }

        }

    }

    cout<< "Generators are: ";

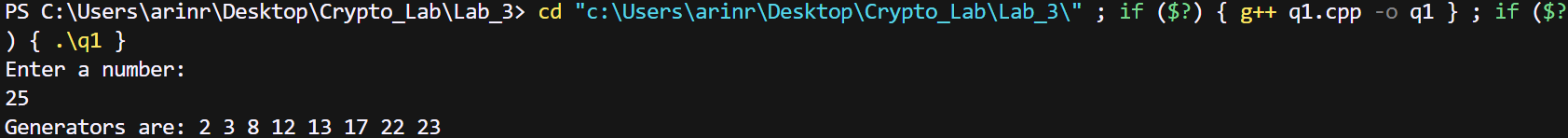
    for(*auto* j : order[phi]){

        cout<< j<< " ";

    }

    cout<<endl;

}

*Output – *

**Q2. Write a program to find the list of cyclic group present within a range (Example: 2000 to 3000).**

**A:**

*Pseudocode and Explanation –*

func bin() – // function used to convert an integer to binary digits

func easy\_mod() - // function used to calculate *ae mod n*  for large integers.

* Function calls bin to convert an integer to binary form. Then apply repeated square and multiply algorithm for exponentiation to calculate *ae mod n* .

func gcd() - // function to find the gcd of two numbers

func if\_cyclic() –

* Takes input as a value n and return if Z­n­\* is cyclic for this n or not.
* We find Z­n­\* and calulate phi(n) i.e. the number of elements of Z­n­\* .
* Next, we calculate all the factors of phi(n).
* Next, the outer loop traverses through all the elements in Z­n­\*. The inner loop traverses through all the elements in factors. Now, I calculated order of each element using a data structure ***map*.** This allowed me to store the different type of order that can occur and their corresponding Z­n­\* value.
* Now, I stored all the generator in a vector.
* If the size of this vector is not equal to 0 then we can infer that Z­­n\*­ ­is cyclic. ­

func main() – this function just asks for the range as an input and calculates all the cyclic Z­n\* in that range.

*Code –*

#include<bits/stdc++.h>

using *namespace* std;

*void* bin(*unsigned* *n*, vector<*int*> *&vec*){

    if (*n* > 1)

        bin(*n* / 2, *vec*);

*int* x = *n* % 2;

*vec*.push\_back(x);

}

*int* easy\_mod(*int* *a*, *int* *e*, *int* *n*){

    vector<*int*> bin\_repr;

    bin(*e*,bin\_repr);

    reverse(bin\_repr.begin(),bin\_repr.end());

    // for(int i=0;i<bin\_repr.size();i++){

    //     cout<<bin\_repr[i]<<" ";

    // }

*int* A = *a*;

*int* b;

    if(bin\_repr[0] == 1){

        b = A;

    }else{

        b = 1;

    }

    for(*int* i=1;i<bin\_repr.size();i++){

        A = (A\*A)%*n*;

        if(bin\_repr[i] == 1){

            b = (A\*b)%*n*;

        }

    }

    // cout<<b;

    return b;

}

*int* gcd(*int* *a*, *int* *b*)

{

    // Find Minimum of a and b

*int* result = min(*a*, *b*);

    while (result > 0) {

        if (*a* % result == 0 && *b* % result == 0) {

            break;

        }

        result--;

    }

    // Return gcd of a and b

    return result;

}

*bool* if\_cyclic(*int* *n*){

    vector<*int*> zn\_star;

    for(*int* i=1;i<=*n*-1;i++){

        if(gcd(i,*n*) == 1){

            zn\_star.push\_back(i);

        }

    }

*int* phi = zn\_star.size();

    // cout<<phi;

    vector<*int*> factors;

    for(*int* i=1;i<=phi;i++){

        if(phi%i == 0){

            factors.push\_back(i);

            // cout<<i<<" ";

        }

    }

    unordered\_map<*int*, vector<*int*>> order;

    for(*int* i=0;i<zn\_star.size();i++){

        for(*int* j=0;j<factors.size();j++){

*int* x = zn\_star[i];

*int* y = factors[j];

*int* val = easy\_mod(x,y,*n*);

            if(val == 1){

                order[y].push\_back(x);

                // cout<< x <<" "<< y<<endl;

                break;

            }

        }

    }

    vector<*int*> gen;

    for(*auto* j : order[phi]){

        gen.push\_back(j);

    }

    if(gen.size()!=0){

        return true;

    }

    return false;

}

*int* main()

{

    vector<*int*> cyc;

*int* start, end;

    cout<<"Enter the start value of the range: "<<endl;

    cin>>start;

    cout<<"Enter the end value of the range: "<<endl;

    cin>>end;

    for(*int* i=start;i<=end;i++){

        if(if\_cyclic(i)){

            cyc.push\_back(i);

        }

    }

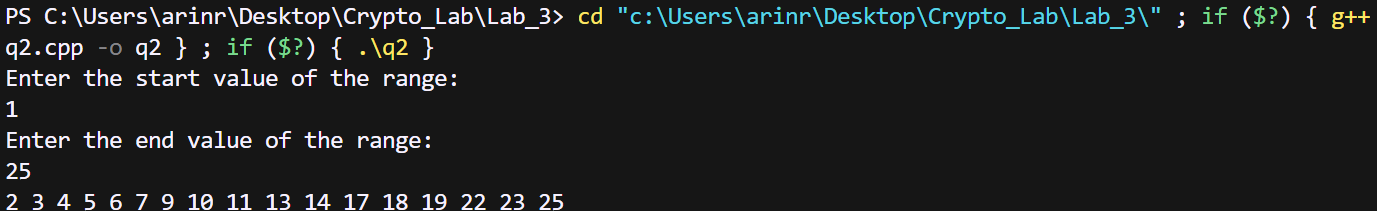
    for(*int* i=0;i<cyc.size();i++){

        cout<<cyc[i]<<" ";

    }

}

*Output-*



**Q3. Write a program to find the order of an element in Zn \*where n is a large integer.**

**A:**

*Pseudocode and Explanation –*

func bin() – // function used to convert an integer to binary digits

func easy\_mod() - // function used to calculate *ae mod n*  for large integers.

* Function calls bin to convert an integer to binary form. Then apply repeated square and multiply algorithm for exponentiation to calculate *ae mod n* .

func gcd() - // function to find the gcd of two numbers

func main() –

* We find Z­n­\* and calulate phi(n) i.e. the number of elements of Z­n­\* .
* Next, we calculate all the factors of phi(n).
* Next, the outer loop traverses through all the elements in Z­n­\*. The inner loop traverses through all the elements in factors. Now, I calculated order of each element using a data structure ***map*.** This allowed me to store the different type of order that can occur and their corresponding Z­n­\* value.
* Lastly I just printed all the value of the map and there corresponding vector values.

*Code –*

#include<bits/stdc++.h>

using *namespace* std;

*void* bin(*unsigned* *n*, vector<*int*> *&vec*){

    if (*n* > 1)

        bin(*n* / 2, *vec*);

*int* x = *n* % 2;

*vec*.push\_back(x);

}

*int* easy\_mod(*int* *a*, *int* *e*, *int* *n*){

    vector<*int*> bin\_repr;

    bin(*e*,bin\_repr);

    reverse(bin\_repr.begin(),bin\_repr.end());

    // for(int i=0;i<bin\_repr.size();i++){

    //     cout<<bin\_repr[i]<<" ";

    // }

*int* A = *a*;

*int* b;

    if(bin\_repr[0] == 1){

        b = A;

    }else{

        b = 1;

    }

    for(*int* i=1;i<bin\_repr.size();i++){

        A = (A\*A)%*n*;

        if(bin\_repr[i] == 1){

            b = (A\*b)%*n*;

        }

    }

    // cout<<b;

    return b;

}

*int* gcd(*int* *a*, *int* *b*)

{

    // Find Minimum of a and b

*int* result = min(*a*, *b*);

    while (result > 0) {

        if (*a* % result == 0 && *b* % result == 0) {

            break;

        }

        result--;

    }

    // Return gcd of a and b

    return result;

}

*int* main()

{

*int* n;

    cout<<"Enter a number: "<<endl;

    cin>>n;

    vector<*int*> zn\_star;

    for(*int* i=1;i<=n-1;i++){

        if(gcd(i,n) == 1){

            zn\_star.push\_back(i);

        }

    }

*int* phi = zn\_star.size();

    // cout<<phi;

    vector<*int*> factors;

    for(*int* i=1;i<=phi;i++){

        if(phi%i == 0){

            factors.push\_back(i);

            // cout<<i<<" ";

        }

    }

    unordered\_map<*int*, vector<*int*>> order;

    for(*int* i=0;i<zn\_star.size();i++){

        for(*int* j=0;j<factors.size();j++){

*int* x = zn\_star[i];

*int* y = factors[j];

*int* val = easy\_mod(x,y,n);

            if(val == 1){

                order[y].push\_back(x);

                cout<< x <<" - "<< y<<endl;

                break;

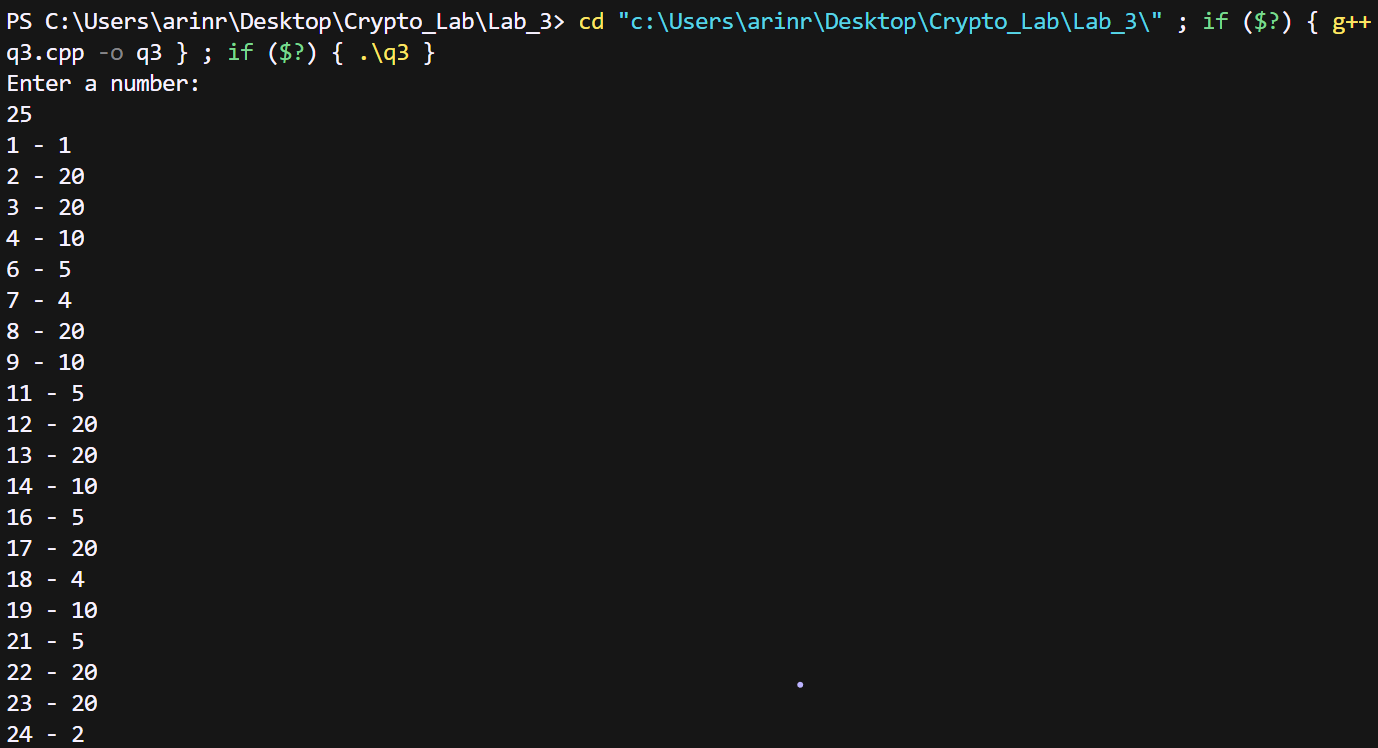
            }

        }

    }

}

*Output –*



**Q4. Write a program to find the quadratic residue and quadratic nonresidue mod n where n is a large integer.**

**A:**

*Pseudocode and Explanation –*

func gcd() - // function to find the gcd of two numbers

func find\_el() – // function to check whether a certain element is present in the vector or not.

Func main() -

* We find Z­n­\*
* For each Z­n­\* value, we calculate the quadratic residue and push it into qn vector.
* The we traverse through the Z­n­\* loop and all those elements that are not present in qn are pushed to qn\_bar vector.

*Code –*

#include<bits/stdc++.h>

using *namespace* std;

*int* gcd(*int* *a*, *int* *b*)

{

    // Find Minimum of a and b

*int* result = min(*a*, *b*);

    while (result > 0) {

        if (*a* % result == 0 && *b* % result == 0) {

            break;

        }

        result--;

    }

    // Return gcd of a and b

    return result;

}

*bool* find\_el(vector<*int*> *vec*, *int* *x*){

    for(*int* i=0;i<*vec*.size();i++){

        if(*vec*[i] == *x*){

            return true;

        }

    }

    return false;

}

*int* main()

{

*int* n;

    cout<<"Enter a number: "<<endl;

    cin>>n;

    vector<*int*> zn\_star;

    for(*int* i=1;i<=n-1;i++){

        if(gcd(i,n) == 1){

            zn\_star.push\_back(i);

        }

    }

    vector<*int*> qn;

    for(*int* i=0;i<zn\_star.size();i++){

*int* x = (zn\_star[i]\*zn\_star[i])%n;

        if(qn.size() == 0 || (!find\_el(qn,x))){

            qn.push\_back(x);

        }

    }

    for(*int* i=0;i<qn.size();i++){

        cout<<qn[i]<<" ";

    }

    cout<<endl;

    vector<*int*> qn\_bar;

    for(*int* i=0;i<zn\_star.size();i++){

*bool* var = false;

        for(*int* j=0;j<qn.size();j++){

            if(zn\_star[i] == qn[j]){

                var = true;

            }

        }

        if(var == false){

            qn\_bar.push\_back(zn\_star[i]);

        }

    }

    for(*int* i=0;i<qn\_bar.size();i++){

        cout<<qn\_bar[i]<<" ";

    }

}

*Output –*



**Q5. Write a program to find the square root of a modulo n where n is a large integer.**

**A:**

*Pseudocode and Explanation –*

func gcd() - // function to find the gcd of two numbers

Func main() -

* We find Z­n­\*
* For each Z­n­\* value, we calculate the square root for each quadratic residue.

*Code –*

#include<bits/stdc++.h>

using *namespace* std;

*int* gcd(*int* *a*, *int* *b*)

{

    // Find Minimum of a and b

*int* result = min(*a*, *b*);

    while (result > 0) {

        if (*a* % result == 0 && *b* % result == 0) {

            break;

        }

        result--;

    }

    // Return gcd of a and b

    return result;

}

*int* main()

{

*int* n;

    cout<<"Enter a number: "<<endl;

    cin>>n;

    vector<*int*> zn\_star;

    for(*int* i=1;i<=n-1;i++){

        if(gcd(i,n) == 1){

            zn\_star.push\_back(i);

        }

    }

    // for(int i=0;i<zn\_star.size();i++){

    //     cout<<zn\_star[i]<<" ";

    // }

    unordered\_map<*int*,vector<*int*>> square\_root;

    for(*int* i=0;i<zn\_star.size();i++){

*int* x = (zn\_star[i]\*zn\_star[i])%n;

        square\_root[x].push\_back(zn\_star[i]);

    }

    // for(auto i:square\_root){

    //     cout<<"Square root of "<<i.first<<" are: ";

    //     for(auto j:square\_root[i.first]){

    //         cout<<j<<" ";

    //     }

    //     cout<<endl;

    // }

*int* a = 12;

    cout<<"Square root of "<<a<<" are: ";

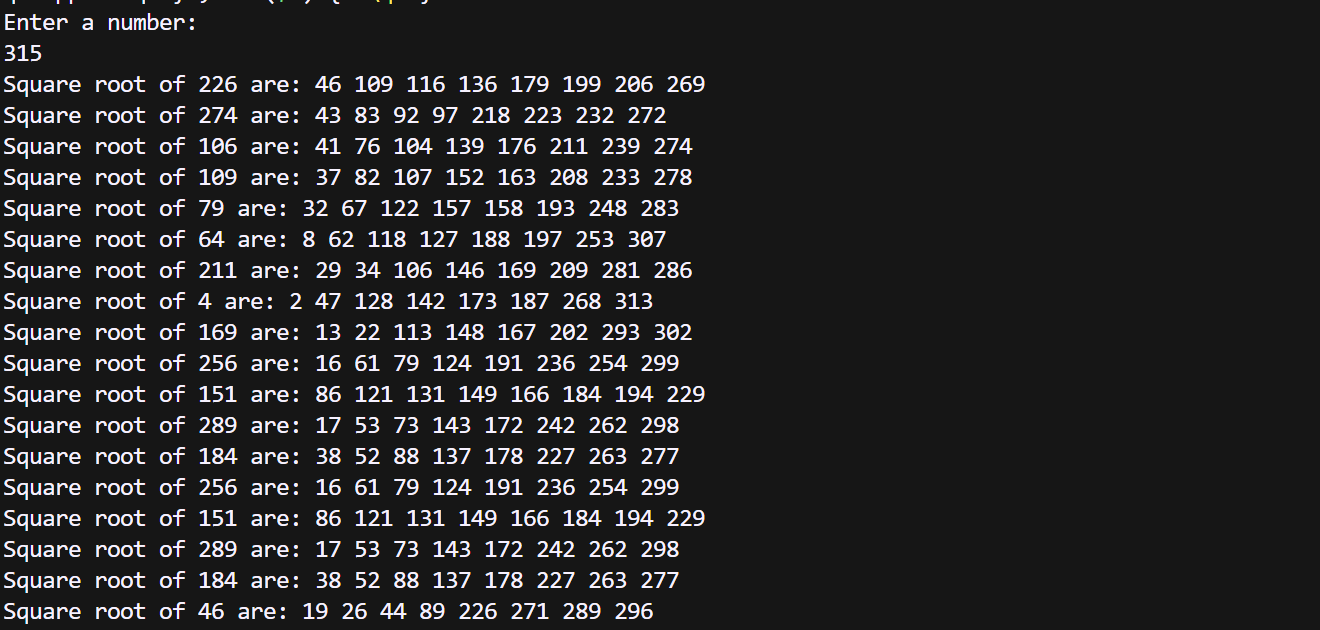
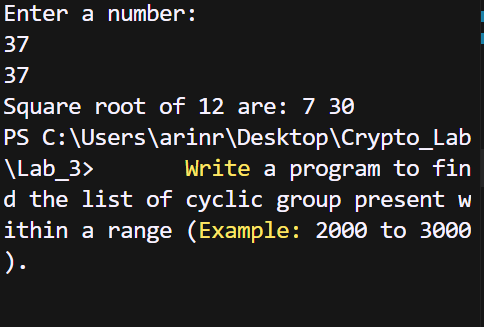
    for(*auto* j:square\_root[a]){

        cout<<j<<" ";

    }

}

*Output –*

** **