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 $a^e \; mod \; n$ for large integers.

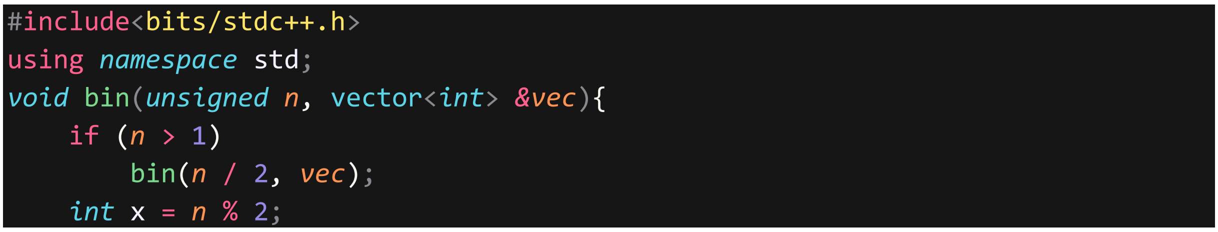
- Function calls bin to convert an integer to binary form. Then apply repeated square and multiply algorithm for exponentiation to calculate $a^e \; mod \; n$.

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

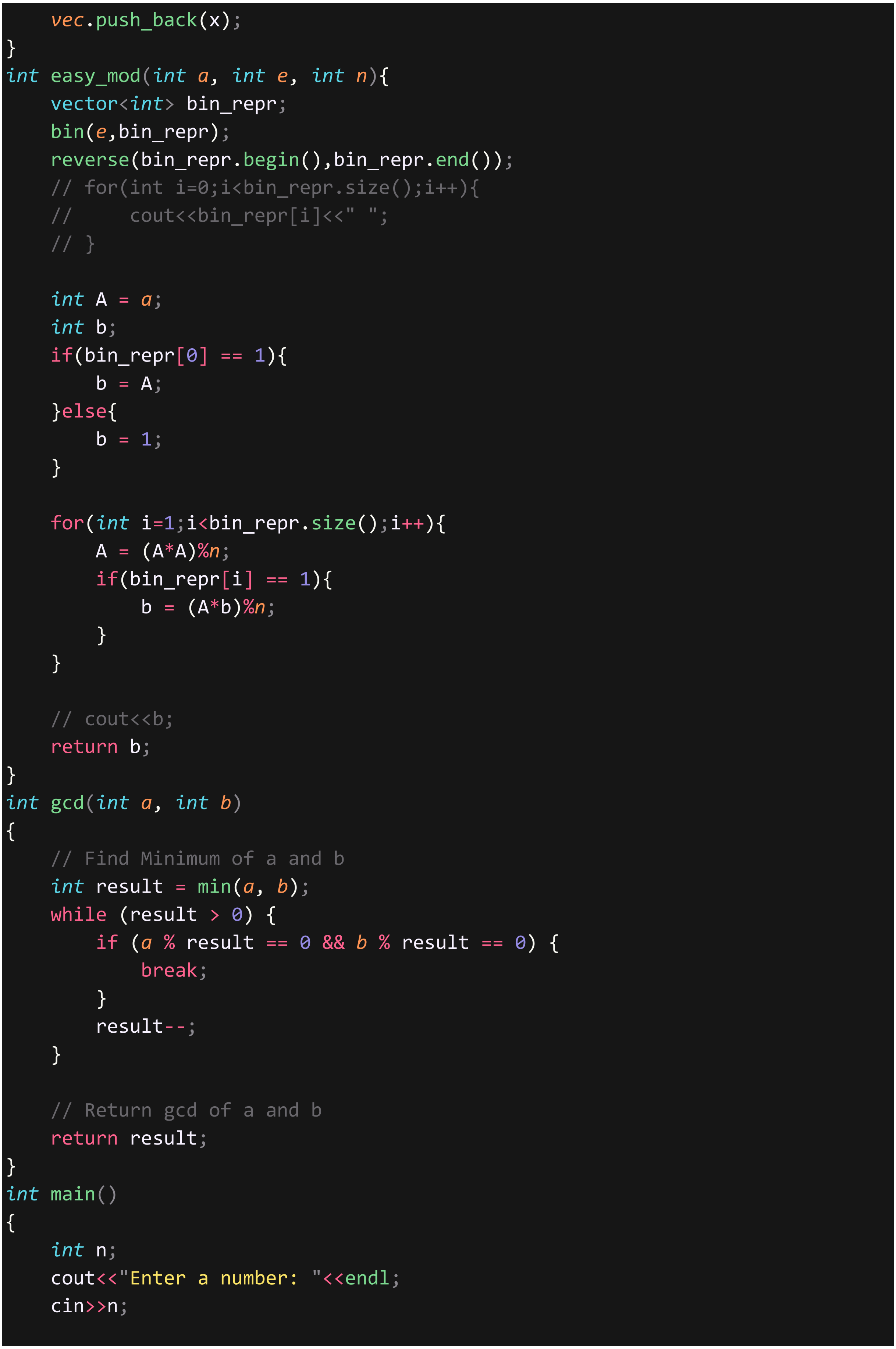
func main() –

- We find Z_n^* and calulate phi(n) i.e. the number of elements of Z_n^* .
- Next, we calculate all the factors of phi(n).
- Next, the outer loop traverses through all the elements in Z_n^*. 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 Z_n^* value.
- Lastly I just printed the map value for phi.

Code -



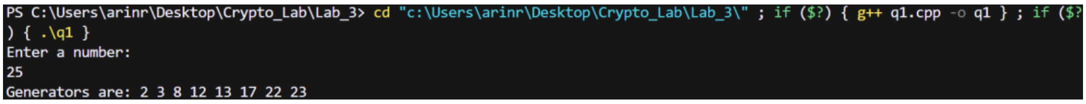
```
#include<bits/stdc++.h>
using namespace std;
void bin(unsigned n, vector<int> &vec){
    if (n > 1)
        bin(n / 2, vec);
    int x = n % 2;
```



```
    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;
```

```cpp
    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 –



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_3> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_3\" ; if ($?) { g++ q1.cpp -o q1 } ; if ($?
) { .\q1 }
Enter a number:
25
Generators are: 2 3 8 12 13 17 22 23
```

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 $a^e \mathit{mod}\ n$ for large integers.

- Function calls bin to convert an integer to binary form. Then apply repeated square and multiply algorithm for exponentiation to calculate $a^e \mathit{mod}\ n$.

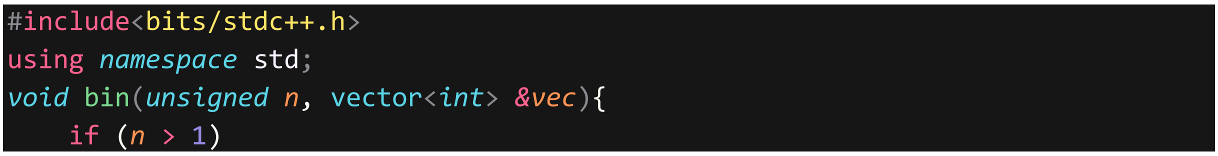
func gcd() - // function to find the gcd of two numbers

func if_cyclic() –

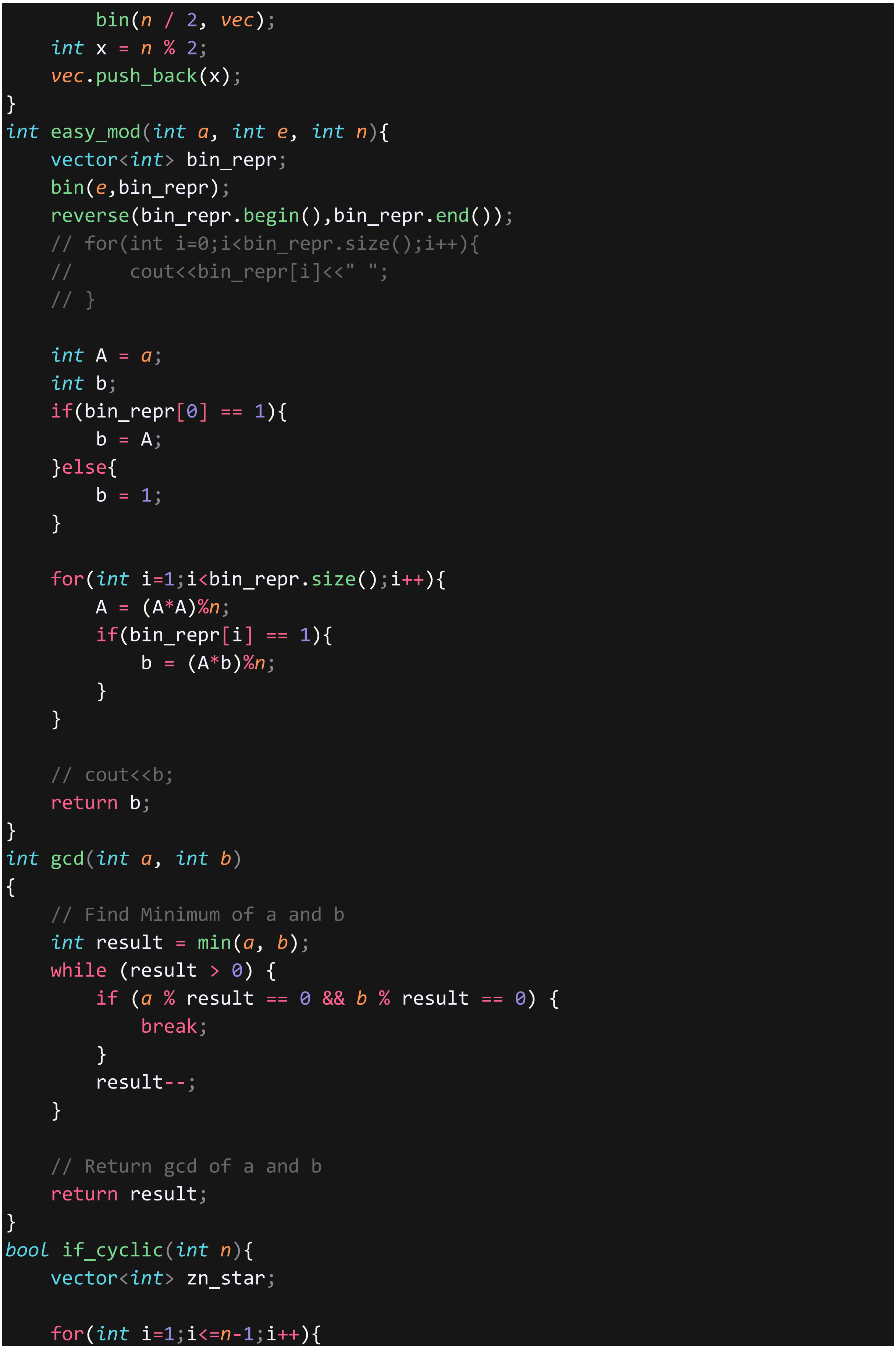
- Takes input as a value n and return if Z_n^* is cyclic for this n or not.
- We find Z_n^* and calulate phi(n) i.e. the number of elements of Z_n^* .
- Next, we calculate all the factors of phi(n).
- Next, the outer loop traverses through all the elements in Z_n^*. The inner loop traverses through all 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 Z_n^* 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 Z_n^* is cyclic.

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

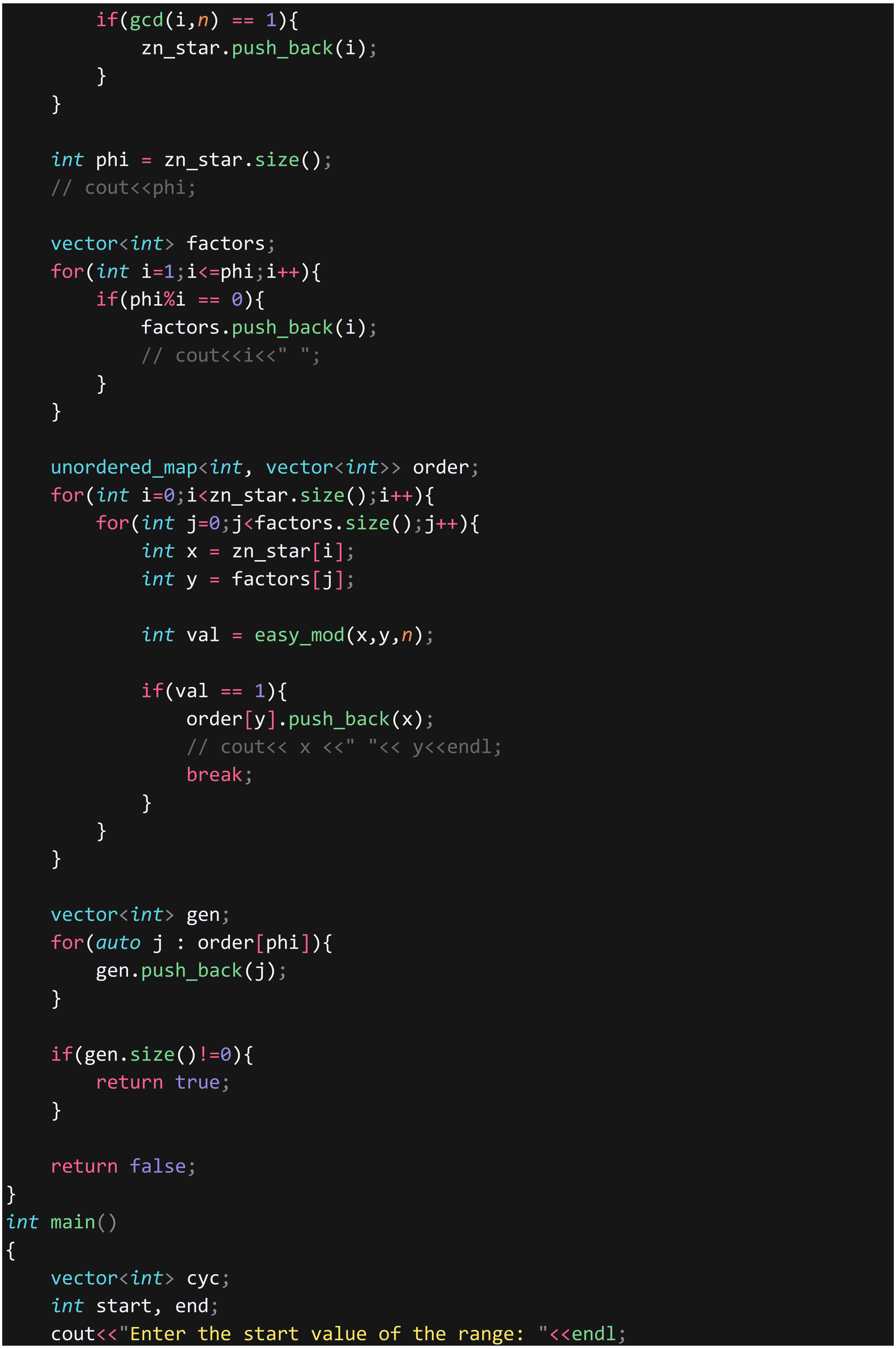
Code –



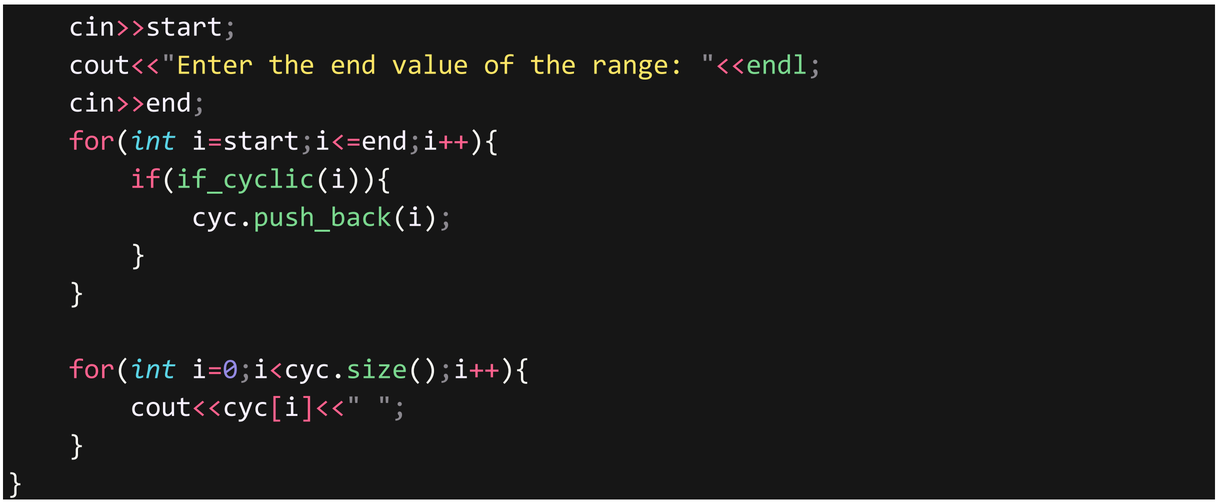
```
#include<bits/stdc++.h>
using namespace std;
void bin(unsigned n, vector<int> &vec){
    if (n > 1)
```



```
        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++){
```

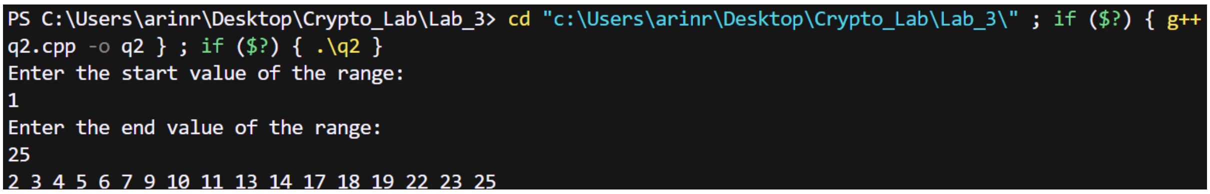


```
        if(gcd(i,n) == 1){
            zn_star.push_back(i);
        }
    }

    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-



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_3> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_3\" ; if ($?) { g++
q2.cpp -o q2 } ; if ($?) { .\q2 }
Enter the start value of the range:
1
Enter the end value of the range:
25
2 3 4 5 6 7 9 10 11 13 14 17 18 19 22 23 25
```

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

A:

<u>*Pseudocode and Explanation –*</u>

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

func easy_mod() - // function used to calculate $a^e \, mod \, n$ for large integers.

- Function calls bin to convert an integer to binary form. Then apply repeated square and multiply algorithm for exponentiation to calculate $a^e \, mod \, n$.

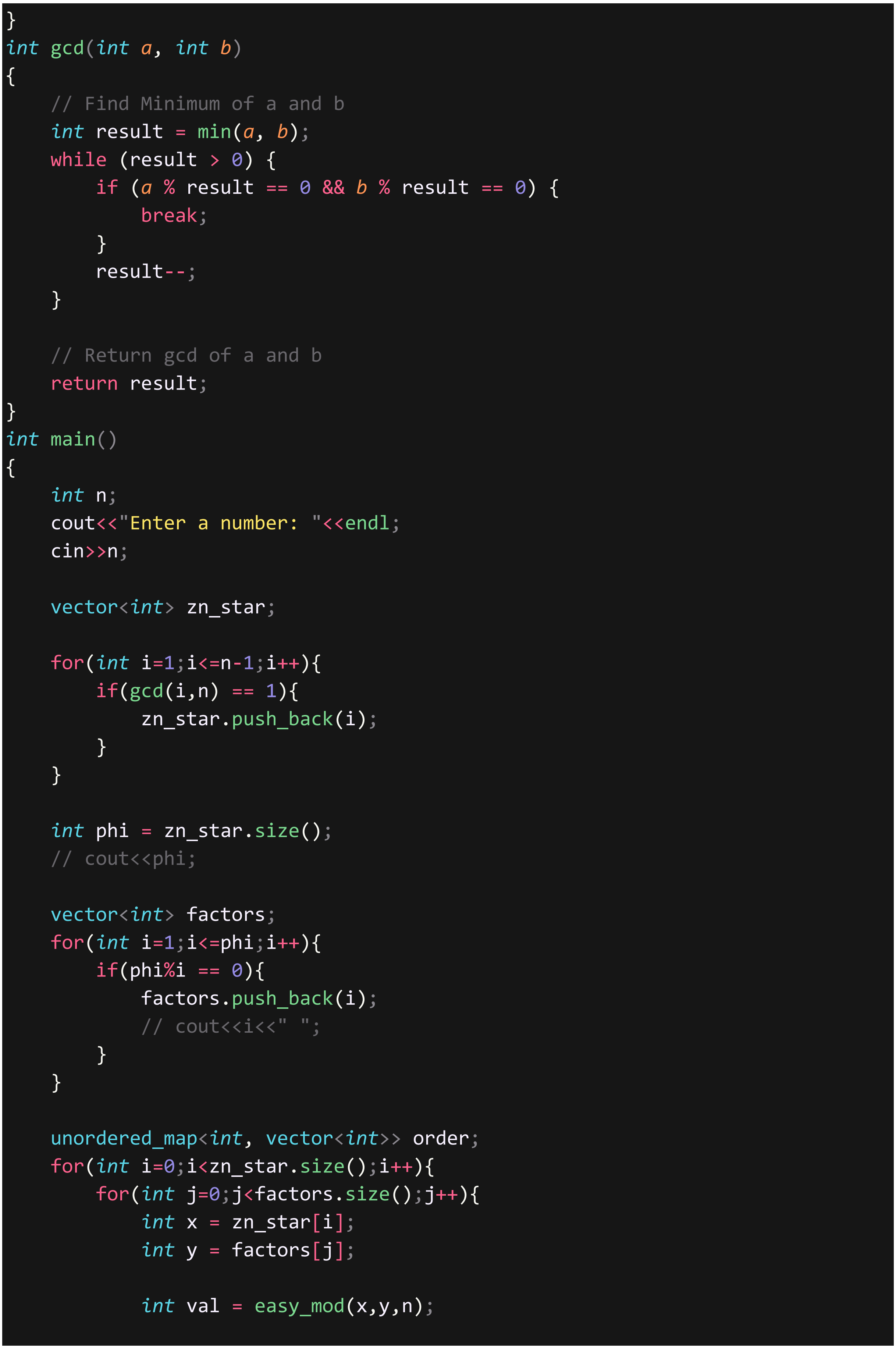
func gcd() - // function to find the gcd of two numbers

func main() –

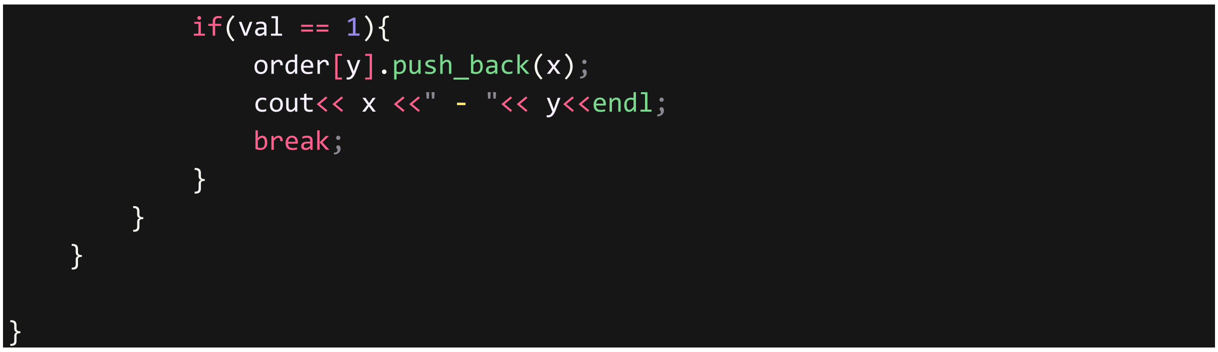
- We find Z_n^* and calulate phi(n) i.e. the number of elements of Z_n^* .
- Next, we calculate all the factors of phi(n).
- Next, the outer loop traverses through all the elements in Z_n^*. The inner loop traverses through all the elements in factors. Now, I calculated order of each element using a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

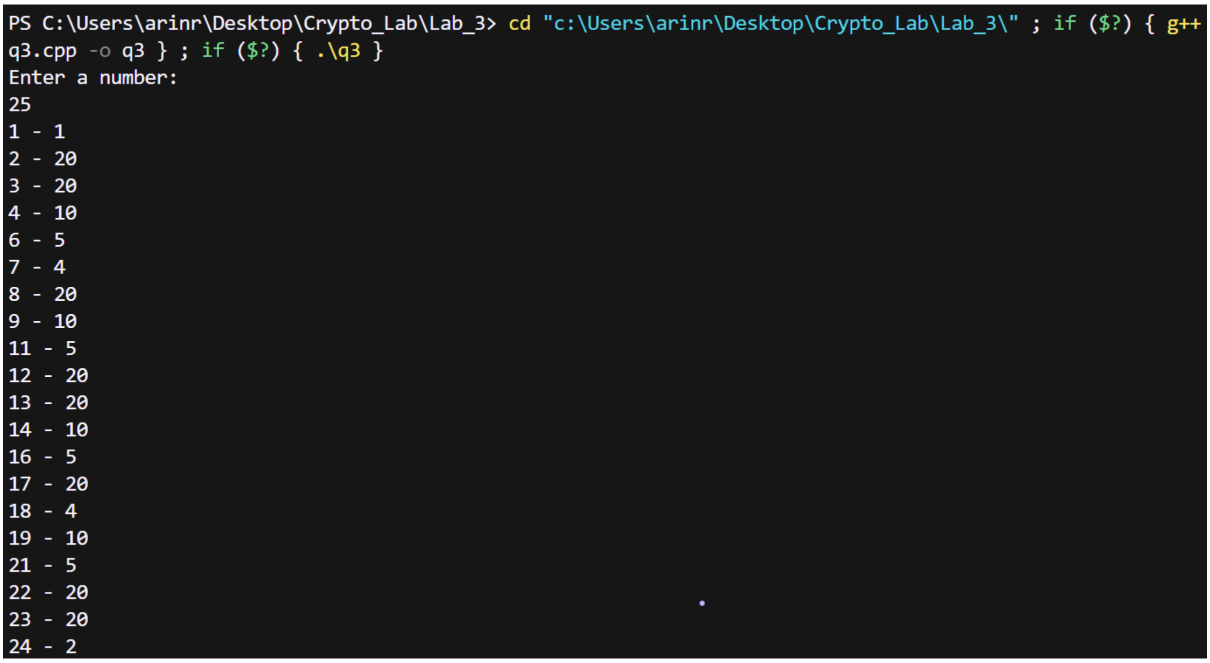


```
}
int gcd(int a, int b)
{
    // Find Minimum of a and b
    int result = min(a, b);
    while (result > 0) {
        if (a % result == 0 && b % result == 0) {
            break;
        }
        result--;
    }

    // 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 –



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_3> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_3\" ; if ($?) { g++
q3.cpp -o q3 } ; if ($?) { .\q3 }
Enter a number:
25
1 - 1
2 - 20
3 - 20
4 - 10
6 - 5
7 - 4
8 - 20
9 - 10
11 - 5
12 - 20
13 - 20
14 - 10
16 - 5
17 - 20
18 - 4
19 - 10
21 - 5
22 - 20
23 - 20
24 - 2
```

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.

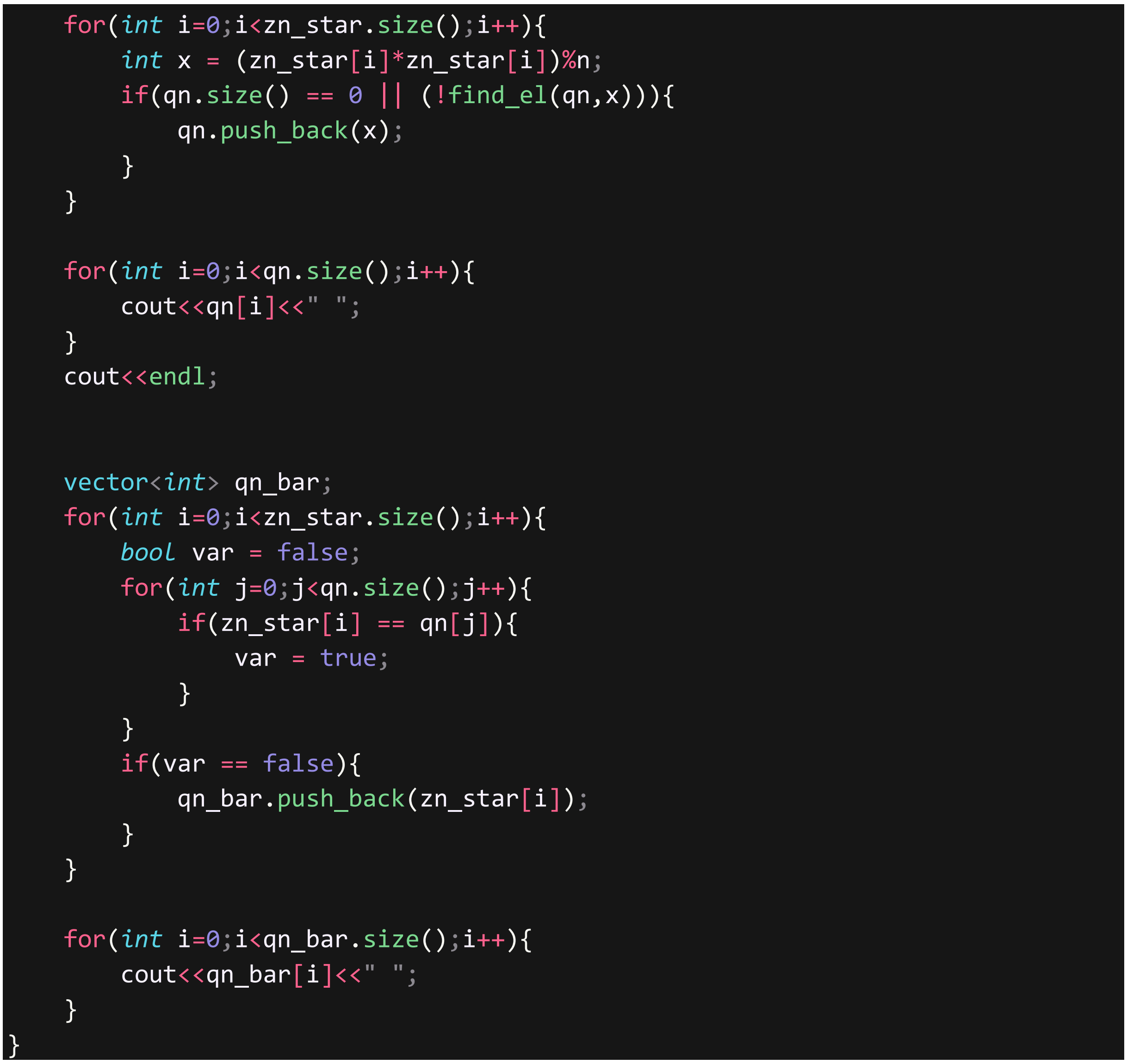
Func main() -

- We find Z_n^*

- For each Z_n^* value, we calculate the quadratic residue and push it into qn vector.
- The we traverse through the Z_n^* loop and all those elements that are not present in qn are pushed to qn_bar vector.

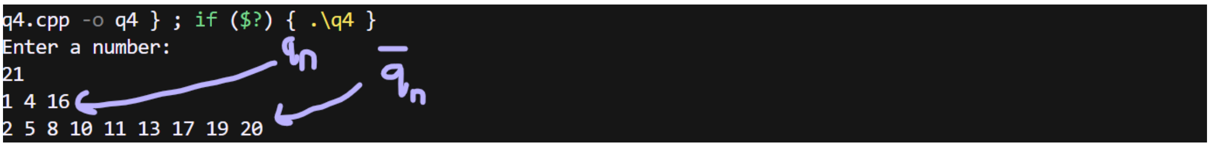
Code –

```
#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 –



```
q4.cpp -o q4 } ; if ($?) { .\q4 }
Enter a number:
21
1 4 16
2 5 8 10 11 13 17 19 20
```

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

A:

Pseudocode and Explanation –

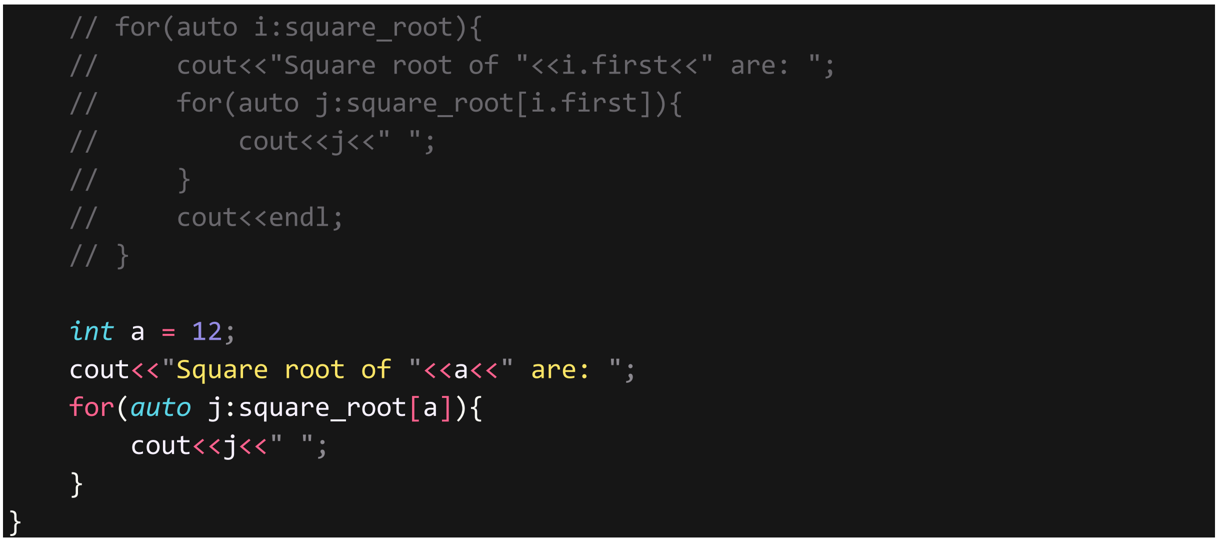
func gcd() - // function to find the gcd of two numbers

Func main() -

- We find Z_n^*
- For each Z_n^* value, we calculate the square root for each quadratic residue.

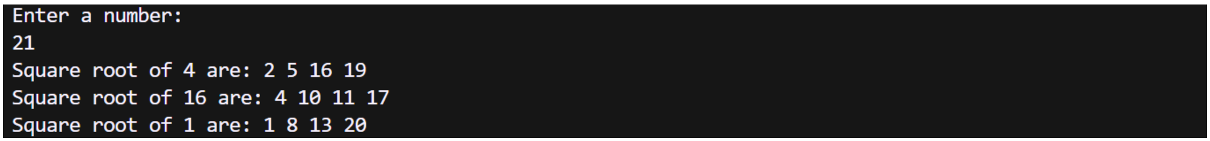
Code –

```
#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]);
    }
```

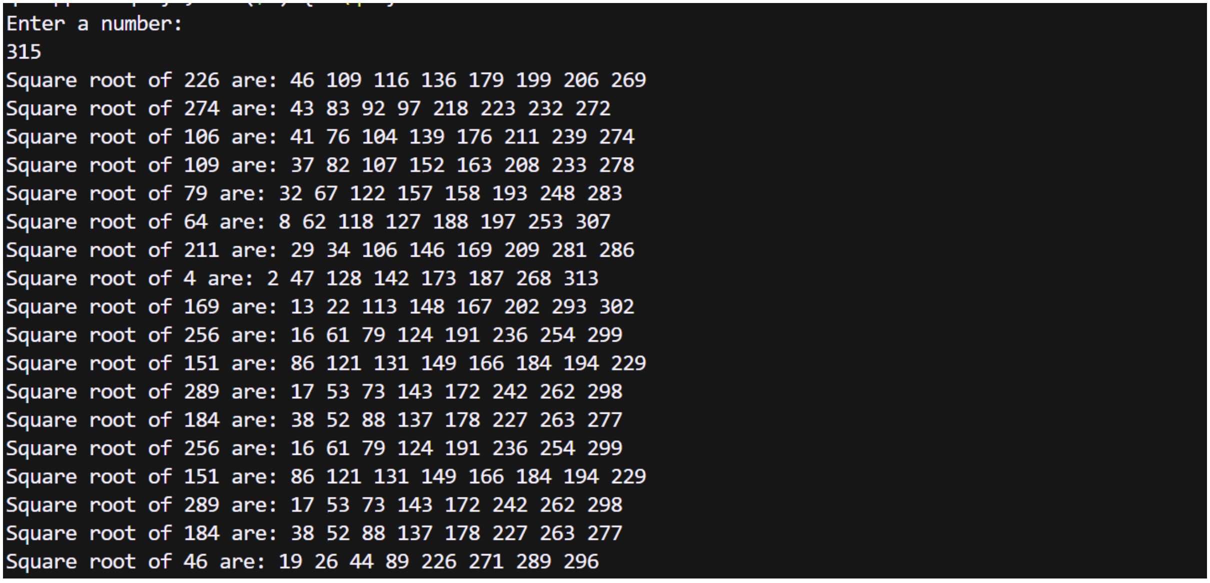


```
    // for(auto i:square_root){
    //     cout<<"Square root of "<<i.first<<" are: ";
    //     for(auto j:square_root[i.first]){
    //         cout<<j<<" ";
    //     }
    //     cout<<endl;
    // }

    int a = 12;
    cout<<"Square root of "<<a<<" are: ";
    for(auto j:square_root[a]){
        cout<<j<<" ";
    }
}
```

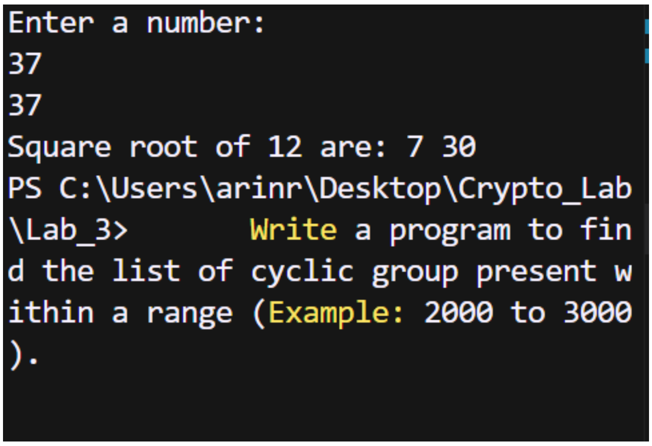
Output –



```
Enter a number:
21
Square root of 4 are: 2 5 16 19
Square root of 16 are: 4 10 11 17
Square root of 1 are: 1 8 13 20
```



```
Enter a number:
315
Square root of 226 are: 46 109 116 136 179 199 206 269
Square root of 274 are: 43 83 92 97 218 223 232 272
Square root of 106 are: 41 76 104 139 176 211 239 274
Square root of 109 are: 37 82 107 152 163 208 233 278
Square root of 79 are: 32 67 122 157 158 193 248 283
Square root of 64 are: 8 62 118 127 188 197 253 307
Square root of 211 are: 29 34 106 146 169 209 281 286
Square root of 4 are: 2 47 128 142 173 187 268 313
Square root of 169 are: 13 22 113 148 167 202 293 302
Square root of 256 are: 16 61 79 124 191 236 254 299
Square root of 151 are: 86 121 131 149 166 184 194 229
Square root of 289 are: 17 53 73 143 172 242 262 298
Square root of 184 are: 38 52 88 137 178 227 263 277
Square root of 256 are: 16 61 79 124 191 236 254 299
Square root of 151 are: 86 121 131 149 166 184 194 229
Square root of 289 are: 17 53 73 143 172 242 262 298
Square root of 184 are: 38 52 88 137 178 227 263 277
Square root of 46 are: 19 26 44 89 226 271 289 296
```



```
Enter a number:
37
37
Square root of 12 are: 7 30
PS C:\Users\arinr\Desktop\Crypto_Lab
\Lab_3>       Write a program to fin
d the list of cyclic group present w
ithin a range (Example: 2000 to 3000
).
```