Lab 2

Name: Yuvraj Singh
Roll No: 23072021
M.Tech CSE

Problem: Write a program to check a given number is prime or not in C++

Algorithm to Check Prime Number:

1. Start
2. Input: Get a positive integer `input_number` from the user.
3. If `input_number` is less than or equal to 1, go to step 11.
4. If `input_number` is equal to 2, go to step 10.
5. If `input_number` is even, go to step 9.
6. Set `is_prime` to true.
7. For `i` from 3 to the square root of `input_number`:
   (a) If `input_number` is divisible evenly by `i` (`input_number % i == 0`), set `is_prime` to false and go to step 9.
   (b) Else set `i := i + 2`.
8. If `is_prime` is true, go to step 10.
9. Output: Print "The number is not prime" and go to step 12.
10. Output: Print "The number is prime" and go to step 12.
11. Output: Print "The number is not prime (since it's less than or equal to 1)" and go to step 12.
12. End

Implementation in C++

```
#include<bits/stdc++.h>
using namespace std;

bool isPrime(int n) {
    if (n<=1) return 0;
    else if (n==2) return 1;
    else if ((n&1)==0) return 0;
    else {
        int i = 3;
        while (i*i <= n) {
            if (n % i == 0) return 0;
            i += 2;
        }
        return true;
    }
}

int main() {
    int t = 10;
    while (t--) {
        int N;
        cout<<"Please␣Enter␣an␣Number␣(N):␣";
        cin>>N;

        if (isPrime(N)) cout<<N<<"␣is␣a␣prime␣number."<<endl;
        else cout<<N<< "␣is␣not␣a␣prime␣number."<<endl;
    }
    return 0;
}
}
```

Time Complexity: O($\sqrt{N}$)

```
input : 61
output : PRIME NUMBER

input : 33
output : NOT A PRIME NUMBER

input : 121
output : NOT A sPRIME NUMBER
```

Problem: Write a program to find prime factorization of a number in C++

Algorithm for Prime Factorization:

1. Start
2. Input: Get a positive integer n from the user.
3. While n is divisible by 2, print 2 and divide n by 2.
4. After step 3, n must be odd. Now start a loop from $i = 3$ to the square root of n. While i divides n, print i and divide n by i, increment i by 2, and continue.
5. If n is a prime number and is greater than 2, then n will not become 1 by the above two steps. So print n if it is greater than 2.
6. End

Example:
Let’s find the prime factorization of the number 84 using the provided algorithm.

1. Start
2. Input: $n = 84$
3. While n is divisible by 2, print 2 and divide n by 2. $(84 \div 2 = 42)$
4. While n is divisible by 2, print 2 and divide n by 2. $(42 \div 2 = 21)$
5. While n is divisible by 3, print 3 and divide n by 3. $(21 \div 3 = 7)$
6. We have reached a prime number (7).
7. The prime factorization of 84 is $2 \times 2 \times 3 \times 7$.
8. So, $84 = 2^2 \times 3 \times 7$.
9. End

Implementation in C++

```

#include<bits/stdc++.h>
using namespace std;

bool isPrime(int n) {
    if (n<=1) return 0;
    else if (n==2) return 1;
    else if ((n&1)==0) return 0;
    else {
        int i = 3;
        while (i*i <= n) {
            if (n % i == 0) return 0;
            i += 2;
        }
        return true;
    }
}

int main() {
    map<int, int> factors;

    int N;
    cout<<"Please␣Enter␣an␣Number␣(N):␣";
    cin>>N;
    if (N==1) {
        cout<<N<<"␣can␣not␣be␣factorized␣because␣it␣is␣neither␣a␣prime␣numbe
        return 0;
    }
    if (isPrime(N)) {
        cout<<N<<"␣is␣a␣prime␣number.␣Therefore␣can␣not␣be␣factorized."<<endl
        return 0;
    }

    while ((N&1) == 0) {
        factors[2]++;
        N>>=1;
    }

    int i =3;
    while (i <= N && !isPrime(N)) {
        if (N % i == 0) {
            factors[i]++;
            N /= i;
            continue;
        }
        i += 2;
    }
    if (N>1) factors[N] = 1;

    string ans = "";
    for (auto x: factors) {
        ans += ("(" + to_string(x.first) + "^" + to_string(x.second) + ")␣*␣
    }

    ans.pop_back(); ans.pop_back();

    cout<<ans<<endl;
    return 0;
}
```

Time Complexity: O($\sqrt{N}$)

input : 84
output : $2^2 \times 3 \times 7$