Lab Report 01

Experiment Name: Implementation of Caesar Cipher

Theory:

The Caesar Cipher is one of the oldest and simplest encryption techniques, named after Julius Caesar who is historically credited with its use. It is a substitution cipher where each letter in the plaintext is shifted a certain number of places down or up the alphabet. The key determines the number of positions each letter is shifted.

History:

The Caesar Cipher has its origins in ancient Rome, where Julius Caesar used it to encrypt his messages. The method provided a basic level of security against casual eavesdropping but is relatively easy to break with modern cryptographic analysis.

Applications of Cryptography:

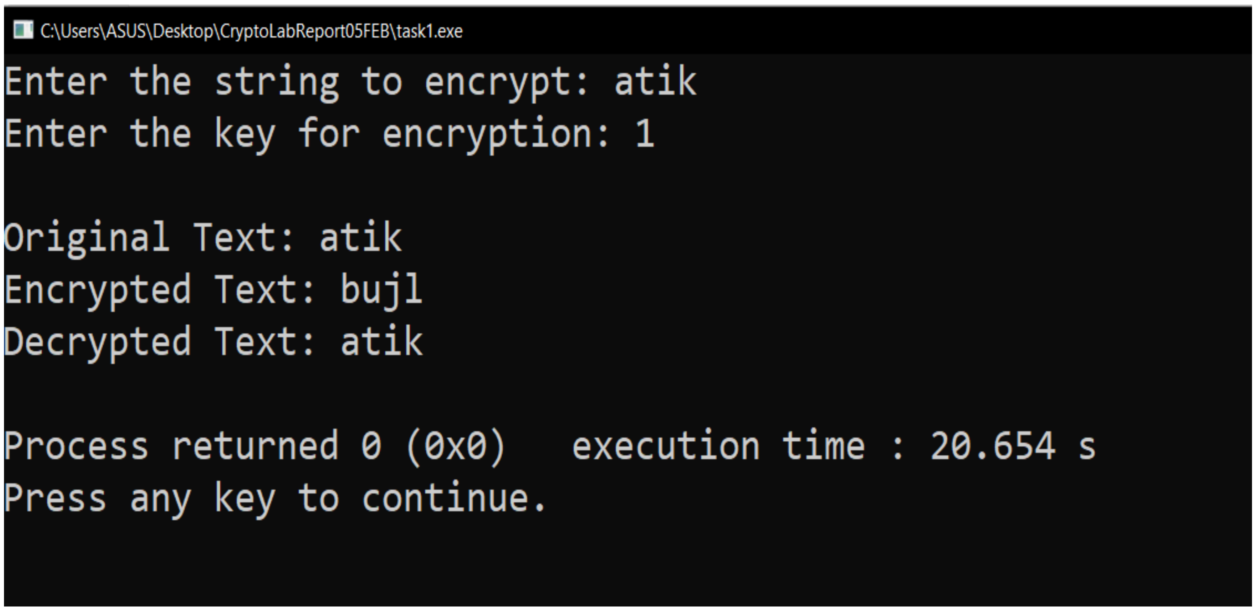
1.Secure Communication
2.Online Banking and Transactions
3.E-commerce Security
4.Password Protection
5.Virtual Private Networks
6.Digital Signatures
7.Secure Email Communication
8.Blockchain Technology
9.Software and Firmware Integrity
10.Military and Government Communications

Code:

```
#include <iostream>
#include <string>
using namespace std;

string encrypt(string text, int key) {
   for (char& c : text) {
      if (isalpha(c)) {
         c = 'a' + (c - 'a' + key) % 26;
      }
   }
   return text;
}

string decrypt(string text, int key) {
   return encrypt(text, 26 - key);
}

int main() {

   cout << "Enter the string to encrypt: ";
   string originalText;
   getline(cin, originalText);

   cout << "Enter the key for encryption: ";
   int key;
   cin >> key;

   for (char& c : originalText) {
      c = tolower(c);
   }

   string encryptedText = encrypt(originalText, key);
   string decryptedText = decrypt(encryptedText, key);
   cout << "\nOriginal Text: " << originalText << endl;
   cout << "Encrypted Text: " << encryptedText << endl;
```

```
    cout << "Decrypted Text: " << decryptedText << endl;
    return 0;
}
```

Output:



```
C:\Users\ASUS\Desktop\CryptoLabReport05FEB\task1.exe
Enter the string to encrypt: atik
Enter the key for encryption: 1

Original Text: atik
Encrypted Text: bujl
Decrypted Text: atik

Process returned 0 (0x0)   execution time : 20.654 s
Press any key to continue.
```

Result and Discussion:

After implementing the Caesar Cipher algorithm in the lab, the results demonstrate a successful encryption and decryption process. The encrypted text is produced by shifting each letter in the plaintext according to the specified key, and decryption is achieved by shifting the letters back. We can use this algorithm for Educational Purposes, Puzzles, and Games.

Conclusion:

In conclusion, the Caesar Cipher is a historic encryption technique that laid the foundation for more advanced cryptographic methods. While it is no longer suitable for secure communication due to its vulnerability, studying the Caesar Cipher provides valuable insights into the principles of encryption and the importance of key management.

Lab Report 02

Experiment Name: Print prime numbers using the Eratosthenes Algorithm.

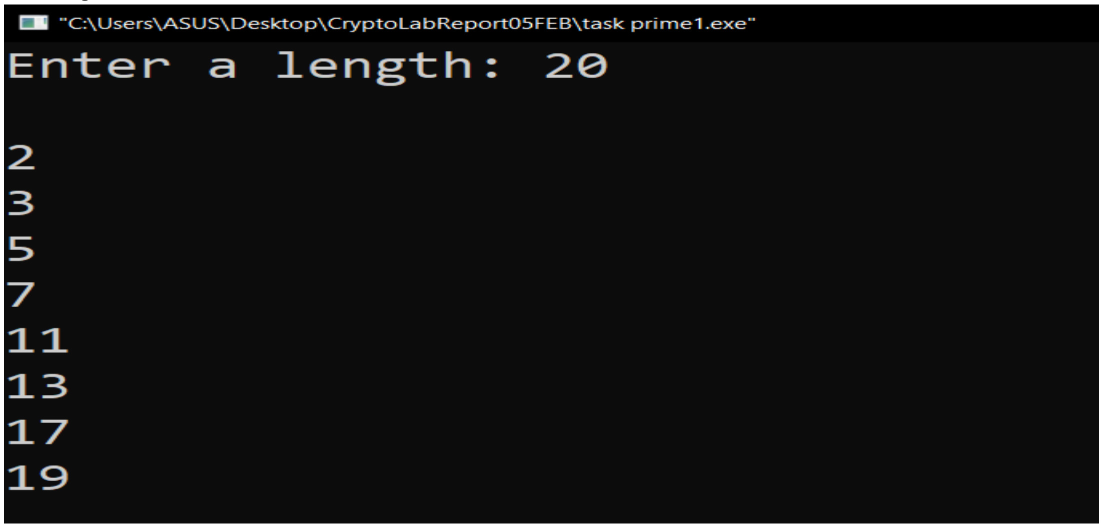
Theory:

The Sieve of Eratosthenes is an efficient algorithm for finding all primes smaller than a given number. It works by iteratively marking the multiples of each prime, starting from the multiples of 2.

Code:

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

int main() {

   int prime[1000]= {0};

   int N;
   cout<<"Enter a length: ";
   cin>>N;
   cout<<endl;

   for(int i=2;i<=N;i++){
      prime[i]=i;
   }

   for(int i=2;i<=N;i+=2){
      if(prime[i] % 2 ==0)
         prime[i]=0;
   }

   prime[2]=2;

   for(int i=3;i<=N;i+=3){
        if(prime[i] % 3==0)
            prime[i]=0;
    }

    prime[3]=3;

    for(int i=5;i<=N;i+=5){
        if(prime[i] % 5 ==0)
            prime[i]=0;
    }

    prime[5]=5;

    for(int i=7;i<=N;i+=7){
        if(prime[i] % 7 ==0)
            prime[i]=0;
    }

    prime[7]=7;

    for(int i=0;i<=N;i++){
            if(prime[i] != 0 )
              cout<<prime[i] <<endl;
    }
}
```

Output:



```
"C:\Users\ASUS\Desktop\CryptoLabReport05FEB\task prime1.exe"
Enter a length: 20

2
3
5
7
11
13
17
19
```

Result and Discussion:
The code takes an input N from the user, initializes an array prime of size 1000, and then applies the Sieve of Eratosthenes algorithm to find all prime numbers up to the given limit N. Finally, it prints the prime numbers. The code initializes an array prime where each index represents a number. Initially, all values are set to the respective index values.
It then iterates through the array starting from 2, marking multiples of each prime number as non-prime (setting them to 0). After completing the Sieve algorithm, the code prints the remaining non-zero values in the array, which correspond to prime numbers.

Conclusion:
The Sieve of Eratosthenes is a simple and efficient algorithm for finding prime numbers up to a given limit. The code successfully implements the algorithm and provides the prime numbers within the specified range. However, it's important to note that the fixed size of the array (prime[1000]) may limit the range of N for which the algorithm can effectively operate. If a larger range is desired, the array size may need to be dynamically allocated or adjusted accordingly. Additionally, the code could benefit from user input validation to ensure a valid input for N.

Lab Report 02

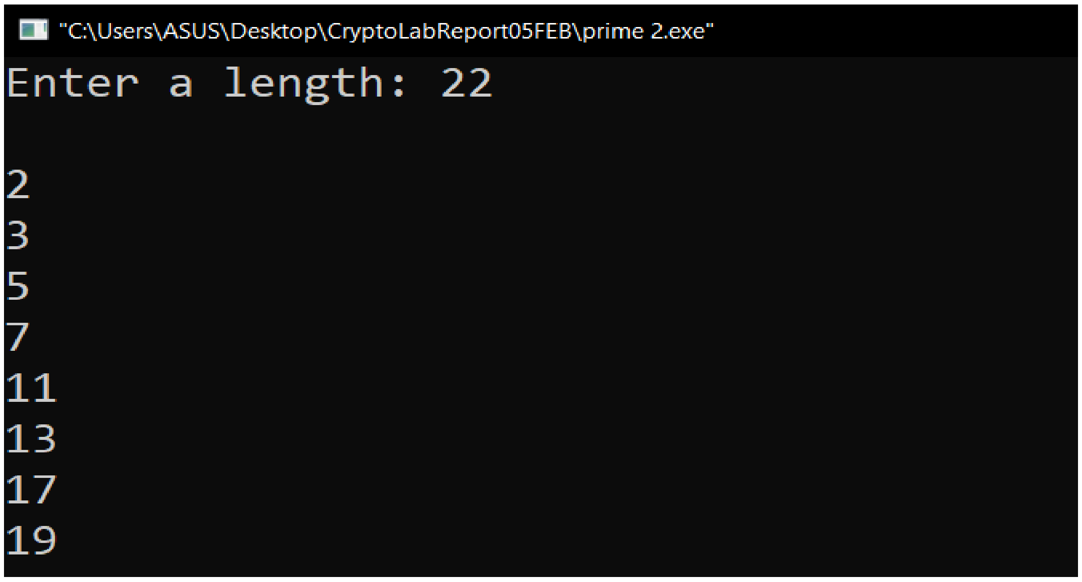
Experiment Name: Extract all the primes found in task 1 into another array called only_primes[N].

Code:

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

int main(){

   int prime[1000]= {0};
   int only_primes[1000] = {0};

   int N;
   cout<<"Enter a length: ";
   cin>>N;
   cout<<endl;

   for(int i=2;i<=N;i++){
      prime[i]=i;
   }

   for(int i=2;i<=N;i+=2){
      if(prime[i] % 2 ==0)
         prime[i]=0;
   }
   prime[2]=2;

   for(int i=3;i<=N;i+=3){
      if(prime[i] % 3==0)
         prime[i]=0;
   }
   prime[3]=3;
```

```
    for(int i=5;i<=N;i+=5){
       if(prime[i] % 5 ==0)
          prime[i]=0;
    }
    prime[5]=5;

    for(int i=7;i<=N;i+=7){
       if(prime[i] % 7 ==0)
          prime[i]=0;
    }
    prime[7]=7;

    for(int i=0;i<=N;i++){
          if(prime[i] != 0 )
            only_primes[i]  = prime[i];
    }

    for(int i=0;i<=N;i++){
          if(only_primes[i] != 0 )
            cout<<only_primes[i] <<endl;
    }
}
```

Output:



```
"C:\Users\ASUS\Desktop\CryptoLabReport05FEB\prime 2.exe"
Enter a length: 22

2
3
5
7
11
13
17
19
```