QUESTION 1 - VIGNERE CIPHER

Vignere.cpp

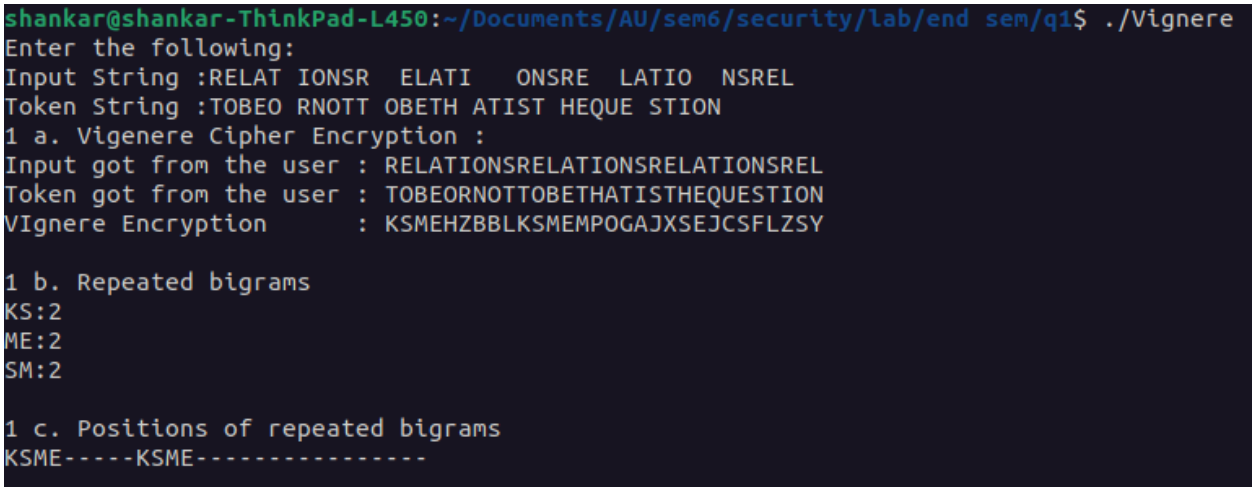
```
#include <iostream>
#include <cstdio>
#include <map>
#include <string>

using namespace std;

//converting the char to integer
int char_to_num(char ch)
{
  return ((int)ch - 65);
}

string remove_spaces(string str)
{
  int i;
  for (i = 0; i < str.size(); i++)
  {
    if (str[i] == ' ')
    {
      str.erase(str.begin() + i);
      i--;
    }
  }
  return str;
}

//counting the bigrams for question b and c
void count_bigrams(string str)
{
  //creating a map and a interator map
  map<string, int> mp;
  map<string, int>::iterator mp_itr;
  string temp_map;

  int i;
```

```
  //converting the string into a map for processing of char - frequency relation
  for (i = 1; i < str.size(); i++)
  {
    temp_map.clear();
    temp_map.push_back(str[i - 1]);
    temp_map.push_back(str[i]);

    if (mp.find(temp_map) == mp.end())
      mp.insert(pair<string, int>(temp_map, 1));
    else
      mp[temp_map]++;
  }

  cout << "1 b. Repeated bigrams\n";

  //checking the map with the char - frequency
  for (mp_itr = mp.begin(); mp_itr != mp.end(); mp_itr++)
  {
    //if the frequency is greater than 1 then we have a bigram
    if ((mp_itr->second) > 1)
      cout << mp_itr->first << ":" << mp_itr->second << endl;
  }

  cout << "\n1 c. Positions of repeated bigrams\n";
  for (i = 1; i < str.size(); i++)
  {
    temp_map.clear();
    temp_map.push_back(str[i - 1]);
    temp_map.push_back(str[i]);

    //parse through the map and if we have a bigram then print the value
    if (mp[temp_map] > 1)
    {
      cout << str[i - 1] << str[i];
      i++;
    }
    else
      cout << "-";
  }
  cout << "\n\n";
}

int main()
```

```
{
  string input;
  string token;
  string output;
  int choice;

  cout << "Enter the following:\nInput String :";
  getline(cin, input);

  cout << "Token String :";
  getline(cin, token);

  //cleaning up the spaces
  input = remove_spaces(input);
  token = remove_spaces(token);

  //input val
  int i = 0;
  int i_val;

  //token val
  int t = 0;
  int t_val;

  while (i < input.size())
  {
    if (input[i] == ' ')
      continue;

    i_val = char_to_num(input[i]);
    t_val = char_to_num(token[t]);

    //vignere cipher
    i_val = i_val + t_val;
    i_val = i_val%26 + 65;

    output.push_back((char)i_val);
    t = (t + 1) % token.size();
    i++;
  }

  cout << "1 a. Vigenere Cipher Encryption : \n";
  cout << "Input got from the user : " << input << "\n";
  cout << "Token got from the user : " << token << "\n";
```

```
  cout << "VIgnere Encryption      : " << output << "\n\n";

  count_bigrams(output);
}
```

Output:



QUESTION 2 - DH Client Server

DH.java

```
package cryptfiles;

import java.io.*;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class DH {
	private static long power(long a, long b, long p) {
		//System.out.println(a+"^"+b+"%"+p);
		if (b == 1)
			return a;
		else
			return (((long) Math.pow(a, b)) % p);
	}
```

```
	//read the data from the file
	public static long getVal(String addr) throws FileNotFoundException{
		File file = new File(addr + ".txt");
		Scanner sc = new Scanner(file);

		sc.useDelimiter("\\Z");

		return(Long.parseLong(sc.next()));
	}

	public static long getRandomVal(long maxVal){
		double val = (double)Math.random()*(maxVal  + 1);
		//System.out.println(val);
		return (long)val;
	}

	public static long generateX(long P){
		long X = getRandomVal(P);
		return(X);
	}

	public static long calculateY(long G, long X, long P) {
		long Y = power(G, X, P);
		//System.out.println("Y : " + Y);
		return (Y);
	}

	public static long calculateK(long Y, long X, long P) {
		long K = power(Y, X, P);
		return (K);
	}
}
```

Client.java

```
import java.io.*;
import java.net.*;
import java.security.KeyStore.SecretKeyEntry;

import java.math.*;
import cryptfiles.DH;

public class Client {
   public static void main(String[] args) throws IOException {
      try {
         //read the data from the files
         DH obj = new DH();
         long Q = obj.getVal("cryptfiles/q");
         long Alpha = obj.getVal("cryptfiles/alpha");

         long Xa = obj.getRandomVal(Q);
         System.out.println("The private key a for Alice:" + Xa);

         long Ya = obj.calculateY(Alpha, Xa, Q);
         System.out.println("Ya:" + Ya);

         System.out.println("Sending Ya to Bob");
         Socket sock = new Socket("localhost", 6666);
         DataOutputStream out = new DataOutputStream(sock.getOutputStream());

         out.writeUTF(String.valueOf(Ya));

         DataInputStream in = new DataInputStream(sock.getInputStream());
         long Yb = Long.parseLong((String) in.readUTF());

         System.out.println("\nGot Yb from Bob");
         long Ka = obj.calculateK(Yb, Xa, Q);
         System.out.println("Ka:" + Ka);

         in.close();
         sock.close();
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

Server.java

```
import java.io.*;
import java.net.*;

import cryptfiles.*;

public class Server {
   public static void processData(String arr[]) {
   }

   public static void main(String[] args) throws IOException {

      try {
         DH obj = new DH();
         long Q = obj.getVal("cryptfiles/q");
         long Alpha = obj.getVal("cryptfiles/alpha");


         ServerSocket servSock = new ServerSocket(6666);
         Socket sock = servSock.accept();

         System.out.println("Got Ya from Alice");
         DataInputStream in = new DataInputStream(sock.getInputStream());
         long Ya =  Long.parseLong((String) in.readUTF());

         long Xb = obj.getRandomVal(Q);
         System.out.println("The private key b for Bob:" + Xb);

         long Yb = obj.calculateY(Alpha, Xb, Q);
         System.out.println("Yb:" + Yb);

         long Kb = obj.calculateK(Ya, Xb, Q);
         System.out.println("Kb:" + Kb);

         DataOutputStream out = new DataOutputStream(sock.getOutputStream());
         System.out.println("\nSending Yb to Alice");
         out.writeUTF(String.valueOf(Yb));

         out.flush();
         out.close();

         servSock.close();
      } catch (Exception e) {
```

```
            System.out.println(e);
        }
    }
}
```

Output: