A Mini Project Report

on

CIPHERING AND DECIPHERING

Course: Object Oriented Programming Lab

Sem: III Sec: CSE-B

By

ASHISH PERALA

1602-19-733-065

&

GOPI KRISHNA KULAKARNI

1602-19-733-069

Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

Ibrahimbagh, Hyderabad-31

2020

Acknowledgement

We whole heartedly thank our sir **Mr.M.S.V Sashi Kumar** for guiding us in making this project successful by always being our side and making us think innovatively throughout the course.

ABSTRACT

Generally, this project is developed for enabling data securities for common people and help them to decrypt the messages sent in ciphered languages and this project aims to help the users to encrypt the actual messages and send it to the other. Overall, our application is to provide privacy to the user’s message.

Language used : JAVA

INTRODUCTION

WHAT IS CIPHERING MEANT FOR?

In Cryptology, Ciphering is nothing but an algorithm for performing encryption of a text in a series of well-defined steps. Ciphering is also said as Encoding in a familiar tone.

WHAT IS DECIPHERING MEANT FOR?

Deciphering in a simple way can be said as Decode a text or message sent which is sent in an encrypted mode. This also requires many algorithm and completely reverse steps of ciphering are applied in deciphering mode.

WORKING

Codes generally substitute different length strings of character in the output, while ciphers generally substitute the same number of characters as are input. There are exceptions and some cipher systems may use slightly more, or fewer, characters when output versus the number that were input. The encrypting procedure is varied depending on the key, which changes the detailed operation of the algorithm.

DESCRIPTION

Generally, the program consists of 2 modes i.e., cipher mode and decipher mode.

Firstly, the user is asked to enter his login credentials or else signup and proceed to cipher or decipher the code. Then in cipher mode the user is asked to select one of the cipher codes to encrypt according to their respective complexities.

If the user opts to use the decipher mode, then the user is asked to enter the text along with key to decrypt and is decrypted with the algorithms used in deciphering accordingly.

- Caesar Cipher
  Each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. To cipher a given text we need an integer value, known as shift which indicates the number of positions each letter of the text has been moved down.
- Vigenere Cipher
  This uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets . At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.
- Hill Cipher
  Hill cipher is a polygraphed substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but

this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

- Rail Fence Cipher

  In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence. When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again. Thus the alphabets of the message are written in a zig-zag manner. After each alphabet has been written, the individual rows are combined to obtain the cipher-text.
- Play Fair Cipher

  It is significantly harder to break since the frequency analysis technique used to break simple substitution ciphers is difficult but still can be used on (25*25) = 625 digraphs rather than 25 monographs which is difficult. Frequency analysis thus requires more cipher text to crack the encryption.

The user can view the ciphering and deciphering history , clear history and also can change the password for his account.

IMPLEMENTATION

```
package sherlock;
import java.util.*;
public class User implements java.io.Serializable{
	String userName;
	String password;
	Stack<String> cipherHistory;
	Stack<String> deCipherHistory;
	User(String userName,String password,Stack<String> 
cipherHistory,Stack<String> deCipherHistory){
		this.userName=userName;
		this.password=password;
		this.cipherHistory=cipherHistory;
		this.deCipherHistory=deCipherHistory;
	}
	User(){
		
	}
	void showCipherHistory() {
		Iterator<String> itr=this.cipherHistory.iterator();
		while(itr.hasNext()) {
			System.out.println(itr.next());
		}
	}
	void showDeCipherHistory() {
		Iterator<String> itr=this.deCipherHistory.iterator();
		while(itr.hasNext()) {
			System.out.println(itr.next());
		}
	}
	void clearCipherHistory() {
		Iterator<String> itr=this.cipherHistory.iterator();
		while(itr.hasNext()) {
			this.cipherHistory.pop();
		}
	}
	void clearDeCipherHistory() {
		Iterator<String> itr=this.deCipherHistory.iterator();
		while(itr.hasNext()) {
			this.deCipherHistory.pop();
		}
	}
	public String toString() {
		return " UserName-"+this.userName+" Password-
"+this.password;
	}
}


package sherlock;
import java.util.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.HashMap;
public class CaesarCipher {
	StringBuffer cipheredData=new StringBuffer();
	StringBuffer deCipheredData=new StringBuffer();
	StringBuffer cipher(String data,short shift ) {
		for(byte a:data.getBytes()) {
			if(a<=90&&a>=65) {
				cipheredData.append((char)((a-
65+shift)%26+65));
			}
			else {
				cipheredData.append((char)((a-
97+shift)%26+97));
			}
		}
		return cipheredData;
	}
	StringBuffer deCipher(String data,short shift) {
		for(byte a:data.getBytes()) {
			if(a<=90&&a>=65) {
				deCipheredData.append((char)((a-65-
shift)%26+65));
			}
			else {
				deCipheredData.append((char)((a-97-
shift)%26+97));
			}
		}
		return deCipheredData;
	}
	public static void CaesarCipherstart(HashMap<String,User>
users,String username) throws Exception
     {
		short s;
	    BufferedReader br= new BufferedReader(new
InputStreamReader(System.in));
	    System.out.println("TO WRITE A CIPHER PRESS 1 TO DECIPHER A TEXT
PRESS 2");
	    int choice=Integer.parseInt(br.readLine());
	    Date d1 = new Date();
	    switch(choice) {
	    case 1:
		System.out.println("ENTER TEXT");
		String text=br.readLine();
		System.out.println("ENTER SHIFT(SHIFT IS NUMBER OF
POSITIONS BY WHICH YOU WANT TO REPLACE THE LETTER)");
		s=(short) Integer.parseInt(br.readLine());
		CaesarCipher temp=new CaesarCipher();
		System.out.println("Encryption: "+temp.cipher(text, s));
		users.get(username).cipherHistory.add("CAESAR CIPHER : FROM
"+text+" TO "+temp.cipheredData.toString()+" ON "+d1);
		break;
	    case 2:
		System.out.println("ENTER TEXT");
		text=br.readLine();
		System.out.println("ENTER SHIFT");
		s=(short) Integer.parseInt(br.readLine());
		temp=new CaesarCipher();
		System.out.println("Decryption: "+temp.deCipher(text, s));
		users.get(username).deCipherHistory.add("CAESAR CIPHER :
FROM "+text+" TO "+temp.deCipheredData.toString()+" ON "+d1);
		break;
	    default:
```

```
                    System.out.println("\nWRONG CHOICE YOU WERE SUPPOSED
TO ENTER 1 OR 2");
            }
        }
}

package sherlock;
import java.util.Date;
import java.util.HashMap;
import java.util.Scanner;

public class RailFenceCipher {
        int numRails;

        public RailFenceCipher(int numRails) {
                this.numRails = numRails;
        }

        String getDecryptedData(String data) {
                char[] decrypted = new char[data.length()];
                int n = 0;
                for(int k = 0 ; k < numRails; k ++) {
                        int index = k;
                        boolean down = true;
                        while(index < data.length() ) {
                                //System.out.println(k + " " + index+ " "+ n
);

                                decrypted[index] = data.charAt(n++);

                                if(k == 0 || k == numRails - 1) {
                                        index = index + 2 * (numRails - 1);
                                }
                                else if(down) {
                                        index = index +  2 * (numRails - k -
1);

                                        down = !down;
                                }
                                else {
                                        index = index + 2 * k;
                                        down = !down;
                                }
                        }
                }
                return new String(decrypted);
        }


        String getEncryptedData(String data) {
                char[] encrypted = new char[data.length()];
                int n = 0;


                for(int k = 0 ; k < numRails; k ++) {
                        int index = k;
                        boolean down = true;
                        while(index < data.length() ) {
                                //System.out.println(k + " " + index+ " "+ n
);

                                encrypted[n++] = data.charAt(index);
                                if(k == 0 || k == numRails - 1) {
                                        index = index + 2 * (numRails - 1);
                                }
                                else if(down) {
                                        index = index +  2 * (numRails - k -
1);
                                        
                                        down = !down;
                                }
                                else {
                                        index = index + 2 * k;
                                        down = !down;
                                }
                        }
                }
                return new String(encrypted);
        }
        
        //alternate way not efficient
        String getEncryptedData2(String data) {
                
                int len = data.length();
                StringBuffer[] sb = new StringBuffer[numRails];
                
                for (int i = 0; i < numRails; i++) {
                        sb[i] = new StringBuffer();
                }
                
                int index = 0;
                int direction = 1;
                
                for (int i = 0; i < data.length(); i++) {
                        
                        sb[index].append(data.charAt(i));
                        index = index + direction;
                        
                        if (index == 0) {
                                direction = 1;
                        } else if (index == numRails) {
                                if(index == 2) {// base case for cipher of
length 2
                                        index = 0;
                                }else {
                                        direction = -1;
                                        index = numRails -2;
                                        
                                }
                        }
                        
                }
                
                for (int i = 1; i < numRails; i++) {
                        sb[0].append(sb[i].toString());
                }
                
                return sb[0].toString();
        }
```

```
	public static void RailFenceCipherstart(HashMap<String,User> 
users,String username) throws Exception
	{
	 Scanner sc = new Scanner(System.in);
	 int choice;
	 System.out.println("1.Cipher\n2.Decipher");
	 choice=sc.nextInt();
	 Date d1 = new Date();
	 if(choice==1)
	 {
		 int depth;
	        System.out.println("Enter the key:");
	        depth=sc.nextInt();
		 System.out.println("Enter the line:");
	        String data =sc.next();
		 RailFenceCipher railFenceCipher = new 
RailFenceCipher(depth);
		 String encrypted =railFenceCipher.getEncryptedData(data);
		 System.out.println("Ciphered As:"+encrypted);
		 users.get(username).cipherHistory.add("RAIL FENCE CIPHER : 
FROM "+data+" TO "+encrypted+" USING KEY:"+depth+" ON "+d1);
	 }
	 else if(choice==2)
	 {
		 int depth;
	        System.out.println("Enter the key:");
	        depth=sc.nextInt();
		 System.out.println("Enter the line:");
	        String data =sc.next();
		 RailFenceCipher railFenceCipher = new 
RailFenceCipher(depth);
		 String decrypted = railFenceCipher.getDecryptedData(data);
		 System.out.println("YOUR MESSAGE:"+decrypted);
		 users.get(username).deCipherHistory.add("RAIL FENCE CIPHER 
: FROM "+data+" TO "+decrypted+" USING KEY:"+depth+" ON "+d1);
	 }
	 else
	 {
		 System.out.println("Put Your Brain Hacker!!");
	 }
	 sc.close();
	}
}


package sherlock;
import java.io.*;
import java.util.HashMap;
import java.util.Stack;
public class Sherlock {
	static User CurrentUser;
	static void login(HashMap<String,User> users) throws Exception {
		System.out.println("ENTER YOUR USER NAME");
		BufferedReader br= new BufferedReader(new 
InputStreamReader(System.in));
		String userName;
		while(true) {
			userName=br.readLine();
			if(!users.containsKey(userName)) {
```

```
                    System.out.println("USERNAME DOES NOT EXISTS
REENTER USERNAME");
                }
                else {
                    CurrentUser.userName=userName;
                    break;
                }
            }
            System.out.println("ENTER YOUR PASSWORD");
            String password;
            while(true) {
                password=br.readLine();

        if(password.compareTo((users.get(CurrentUser.userName)).password)
!=0) {
                    System.out.println("WRONG PASSWORD PLEASE
REENTER PASSWORD");
                }
                else {
                    System.out.println("LOGIN SUCCESSFUL...WELCOME
BACK");

                    CurrentUser.password=password;
                    break;
                }
            }
        }
        static void signUp(HashMap<String,User> users) throws Exception {
            Stack<String> h=new Stack<>();
            Stack<String> h1=new Stack<>();
            System.out.println("ENTER USER NAME");
            BufferedReader br= new BufferedReader(new
InputStreamReader(System.in));
            String userName;
            while(true) {
                userName=br.readLine();
                if(users.containsKey(userName)) {
                    System.out.println("USERNAME ALREADY EXISTS
PLEASE TRY ANOTHER ONE");
                }
                else {
                    CurrentUser.userName=userName;
                    break;
                }
            }
            System.out.println("ENTER YOUR PASSWORD");
            String password=br.readLine();
            CurrentUser.password=password;
            User CurrentUserObj=new User(userName,password, h,h1);
            users.put(CurrentUserObj.userName, CurrentUserObj);
            try
         {
           FileOutputStream f = new
FileOutputStream("C:\\Users\\ashup\\eclipse-
workspace\\classwork\\src\\classwork\\userinfo.txt");
           ObjectOutputStream out = new ObjectOutputStream(f);
           // Method for serialization of object
           out.writeObject(users);
           out.close();
           f.close();
```

```
            //System.out.println("Object has been serialized");
        }
        catch(IOException ex)
        {
            System.out.println("IOException is caught");
        }
             //CurrentUser.history=new Stack<String>();
      }
      static void changePassword(HashMap<String,User> users) throws
Exception{
             System.out.println("ENTER YOUR CURRENT PASSWORD");
             String password;
             BufferedReader br= new BufferedReader(new
InputStreamReader(System.in));
             while(true) {
                    password=br.readLine();

      if(password.compareTo((users.get(CurrentUser.userName)).password)
!=0) {
                           System.out.println("WRONG PASSWORD PLEASE
REENTER PASSWORD");
                    }
                    else {
                           System.out.println("ENTER YOUR NEW PASSWORD");
                           password=br.readLine();

      users.get(CurrentUser.userName).password=password;
                           CurrentUser.password=password;
                           System.out.println("PASSWORD CHANGED ");
                           break;
                    }
             }
      }
      public static void main(String[] args) throws Exception {
             CurrentUser=new User();
             System.out.println("TO LOGIN PRESS 1 TO SIGNUP PRESS 2");
             BufferedReader br= new BufferedReader(new
InputStreamReader(System.in));
             int choice=Integer.parseInt(br.readLine());
             HashMap<String,User> users=new HashMap<>();
             try
        {
            FileInputStream file = new
FileInputStream("C:\\Users\\ashup\\eclipse-
workspace\\classwork\\src\\classwork\\userinfo.txt");
            ObjectInputStream in = new ObjectInputStream(file);
            users= (HashMap<String,User>) in.readObject();
            in.close();
            file.close();
        }
        catch(IOException ex)
        {
            System.out.println("IOException is caught..........");
        }
             if(choice==1) {
                    Login(users);
             }
             else if(choice==2) {
                    signUp(users);
                    //System.out.println(users);

        //System.out.println(CurrentUser.userName+CurrentUser.password);
                }
                else {
                        System.out.println("ENTER CORRECT CHOICE");
                        main(null);
                }
                do {
                        System.out.println("CHOOSE YOUR MODE OF ACTION AND 
PRESS RESPECTIVE NUMBERS:\n1.ENTER ENCRYPTION AND DECRTPTION 
MODE\n2.CHECK YOUR PREVIOUS HISTORY\n3.CLEAR YOUR HISTORY\n4.CHANGE YOUR 
PASSWORD\n5.LOG OUT");
                        choice=Integer.parseInt(br.readLine());
                        switch(choice) {
                        case 1:
                                System.out.println("WHICH METHOD DO YOU LIKE 
TO USE\n1.CAESAR CIPHER\n2.HILL CIPHER\n3.PLAY FAIR CIPHER\n4.RAIL FENCE 
CIPHER\n5.VIGENERE CIPHER");

                                int choice2=Integer.parseInt(br.readLine());
                                switch(choice2) {
                                case 1:

        CaesarCipher.CaesarCipherstart(users,CurrentUser.userName);
                                        break;
                                case 2:

        HillCipher.HillCipherStart(users,CurrentUser.userName);
                                        break;
                                case 3:

        PlayFairCipher.playFairCipherstart(users,CurrentUser.userName);
                                        break;
                                case 4:

        RailFenceCipher.RailFenceCipherstart(users,CurrentUser.userName);
                                        break;
                                case 5:

        VigenereCipher.VigenereCipherstart(users,CurrentUser.userName);
                                        break;
                                default:
                                        System.out.println("\nWRONG CHOICE YOU 
WERE SUPPOSED TO ENTER A NUMBER BELOW 5");
                                }
                                break;
                        case 2:
                                System.out.println("WHAT DO YOU LIKE TO 
SEE:\n1.CIPHER MODE HISTORY\n2.DECIPHER MODE HISTORY");
                                choice2=Integer.parseInt(br.readLine());
                                if(choice2==1) {

        users.get(CurrentUser.userName).showCipherHistory();
                                }
                                else if(choice2==2) {

        users.get(CurrentUser.userName).showDeCipherHistory();
                                }
```

```
                          else {
                                System.out.println("WRONG CHOICE YOU
WERE SUPPOSED TO ENTER 1 OR 2");
                          }
                          break;
                    case 3:
                          System.out.println("WHAT HISTORY DO YOU LIKE
TO CLEAR:\n1.CIPHER MODE HISTORY\n2.DECIPHER MODE HISTORY");
                          choice2=Integer.parseInt(br.readLine());
                          if(choice2==1) {

      users.get(CurrentUser.userName).clearCipherHistory();
                          }
                          else if(choice2==2) {

      users.get(CurrentUser.userName).clearDeCipherHistory();
                          }
                          else {
                                System.out.println("WRONG CHOICE YOU
WERE SUPPOSED TO ENTER 1 OR 2 ");
                          }
                          break;
                    case 4:
                          System.out.println(users);
                          changePassword(users);
                          break;
                    case 5:
                          System.out.println("LOG OUT SUCCESSFUL");
                          break;
                    default:
                          System.out.println("\nENTER CORRECT NUMBER");
                    }
             } while(choice!=5);
             try
           {
                  FileOutputStream f = new
FileOutputStream("C:\\Users\\ashup\\eclipse-
workspace\\classwork\\src\\classwork\\userinfo.txt");
                  ObjectOutputStream out = new ObjectOutputStream(f);
                  // Method for serialization of object
                  out.writeObject(users);
                  out.close();
                  f.close();
                  //System.out.println("Object has been serialized");
            }
            catch(IOException ex)
            {
                  System.out.println("IOException is caught");
            }
       }

}


package sherlock;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Date;
```

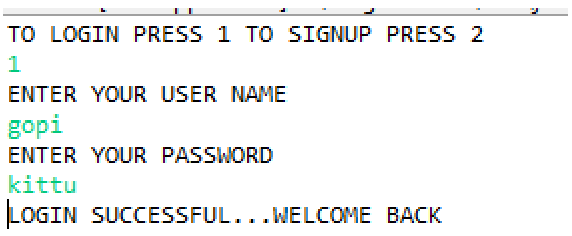
```java
import java.util.HashMap;

public class HillCipher
{
       static HashMap<String,User> users;
    static String username;
    int keymatrix[][];
    int linematrix[];
    int resultmatrix[];

    public void divide(String temp, int s)
    {
        while (temp.length() > s)
        {
            String sub = temp.substring(0, s);
            temp = temp.substring(s, temp.length());
            perform(sub);
        }
        if (temp.length() == s)
            perform(temp);
        else if (temp.length() < s)
        {
            for (int i = temp.length(); i < s; i++)
                temp = temp + 'x';
            perform(temp);
        }
    }

    public void perform(String line)
    {
        linetomatrix(line);
        linemultiplykey(line.length());
        result(line.length());
    }

    public void keytomatrix(String key, int len)
    {
        keymatrix = new int[len][len];
        int c = 0;
        for (int i = 0; i < len; i++)
        {
            for (int j = 0; j < len; j++)
            {
                keymatrix[i][j] = ((int) key.charAt(c)) - 97;
                c++;
            }
        }
    }

    public void linetomatrix(String line)
    {
        linematrix = new int[line.length()];
        for (int i = 0; i < line.length(); i++)
        {
            linematrix[i] = ((int) line.charAt(i)) - 97;
        }
    }

    public void linemultiplykey(int len)
    {
        resultmatrix = new int[len];
        for (int i = 0; i < len; i++)
        {
            for (int j = 0; j < len; j++)
            {
                resultmatrix[i] += keymatrix[i][j] * linematrix[j];
            }
            resultmatrix[i] %= 26;
        }
    }

    public void result(int len)
    {
        String result = "";
        for (int i = 0; i < len; i++)
        {
            result += (char) (resultmatrix[i] + 97);
        }
        System.out.print(result);
        Date d1 = new Date();
        String temp=users.get(username).cipherHistory.pop();
        temp+=result+" ON "+d1;
        users.get(username).cipherHistory.push(temp);
    }

    public boolean check(String key, int len)
    {
        keytomatrix(key, len);
        int d = determinant(keymatrix, len);
        d = d % 26;
        if (d == 0)
        {
            System.out.println("Invalid key!!! Key is not invertible
because determinant=0...");
            return false;
        }
        else if (d % 2 == 0 || d % 13 == 0)
        {
            System.out.println("Invalid key!!! Key is not invertible
because determinant has common factor with 26...");
            return false;
        }
        else
        {
            return true;
        }
    }

    public int determinant(int A[][], int N)
    {
        int res;
        if (N == 1)
            res = A[0][0];
        else if (N == 2)
        {
            res = A[0][0] * A[1][1] - A[1][0] * A[0][1];
        }
        else
        {
            res = 0;
            for (int j1 = 0; j1 < N; j1++)
            {
                int m[][] = new int[N - 1][N - 1];
                for (int i = 1; i < N; i++)
                {
                    int j2 = 0;
                    for (int j = 0; j < N; j++)
                    {
                        if (j == j1)
                            continue;
                        m[i - 1][j2] = A[i][j];
                        j2++;
                    }
                }
                res += Math.pow(-1.0, 1.0 + j1 + 1.0) * A[0][j1]
                        * determinant(m, N - 1);
            }
        }
        return res;
    }

    public void cofact(int num[][], int f)
    {
        int b[][], fac[][];
        b = new int[f][f];
        fac = new int[f][f];
        int p, q, m, n, i, j;
        for (q = 0; q < f; q++)
        {
            for (p = 0; p < f; p++)
            {
                m = 0;
                n = 0;
                for (i = 0; i < f; i++)
                {
                    for (j = 0; j < f; j++)
                    {
                        b[i][j] = 0;
                        if (i != q && j != p)
                        {
                            b[m][n] = num[i][j];
                            if (n < (f - 2))
                                n++;
                            else
                            {
                                n = 0;
                                m++;
                            }
                        }
                    }
                }
                fac[q][p] = (int) Math.pow(-1, q + p) * determinant(b, f
- 1);
            }
        }
        trans(fac, f);
    }
```

```java
void trans(int fac[][], int r)
{
    int i, j;
    int b[][], inv[][];
    b = new int[r][r];
    inv = new int[r][r];
    int d = determinant(keymatrix, r);
    int mi = mi(d % 26);
    mi %= 26;
    if (mi < 0)
        mi += 26;
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            b[i][j] = fac[j][i];
        }
    }
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            inv[i][j] = b[i][j] % 26;
            if (inv[i][j] < 0)
                inv[i][j] += 26;
            inv[i][j] *= mi;
            inv[i][j] %= 26;
        }
    }
    System.out.println("\nIn case if you decipher,use this key:");
    matrixtoinvkey(inv, r);
}

public int mi(int d)
{
    int q, r1, r2, r, t1, t2, t;
    r1 = 26;
    r2 = d;
    t1 = 0;
    t2 = 1;
    while (r1 != 1 && r2 != 0)
    {
        q = r1 / r2;
        r = r1 % r2;
        t = t1 - (t2 * q);
        r1 = r2;
        r2 = r;
        t1 = t2;
        t2 = t;
    }
    return (t1 + t2);
}

public void matrixtoinvkey(int inv[][], int n)
{
    String invkey = "";
    for (int i = 0; i < n; i++)
    {
            for (int j = 0; j < n; j++)
            {
                invkey += (char) (inv[i][j] + 97);
            }
        }
        System.out.print(invkey);
    }
    public static void HillCipherStart(HashMap<String,User> users,String
username) throws Exception
    {
      HillCipher.users=users;
      HillCipher.username=username;
        HillCipher obj = new HillCipher();
        System.out.println("IF LENGTH OF LINE IS n THEN KEY LENGTH
SHOULD BE nxn");
        BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
        int choice;
        System.out.println("1:Cipher\n2:Decipher");
        choice=Integer.parseInt(in.readLine());
        System.out.println("Line: ");
        String line = in.readLine();
        System.out.println("Key:");
        String key = in.readLine();
        double sq = Math.sqrt(key.length());
        Date d1 = new Date();
        if (sq != (long) sq)
            System.out.println("Invalid key length!!! Does not form a
square matrix...");
        else
        {
            int s = (int) sq;
            if (obj.check(key, s))
            {
                System.out.println("Encrypted Message:");
                users.get(username).cipherHistory.add("HILL CIPHER :
FROM "+line+" TO ");
                obj.divide(line, s);
                obj.cofact(obj.keymatrix, s);
            }
        }
    }
}

package sherlock;
import java.util.Date;
import java.util.HashMap;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
      public class VigenereCipher
      {
            public static String cipher(String data, final String
keyword)
          {
              String resultant_string = "";
              data = data.toUpperCase();//converting given cipher to
UpperCase for feasibility//
```

```
                for (int i = 0, j = 0; i < data.length(); i++)
                {
                    char c = data.charAt(i);
                    if (c < 'A' || c > 'Z')
                        continue;
                    resultant_string = resultant_string +(char) ((c + 
keyword.charAt(j) - 2 * 'A') % 26 + 'A');
                    j = ++j % keyword.length();
                }
                return resultant_string;
            }

            public static String decipher(String data, final String 
keyword)
            {
                String resultant_string = "";
                data = data.toUpperCase();
                for (int i = 0, j = 0; i < data.length(); i++)
                {
                    char c = data.charAt(i);
                    if (c < 'A' || c > 'Z')
                        continue;
                    resultant_string = resultant_string+(char) ((c - 
keyword.charAt(j) + 26) % 26 + 'A');
                    j = ++j % keyword.length();
                }
                return resultant_string;
            }

           public static void VigenereCipherstart(HashMap<String,User> 
users,String username) throws Exception
            {
                    Scanner sc=new Scanner(System.in);
                    int choice;
                    System.out.println("1.Cipher\n2.Decipher");
                    choice=sc.nextInt();
                    Date d1 = new Date();
                    if(choice==1)
                    {
                        BufferedReader read=new BufferedReader(new 
InputStreamReader(System.in));
                        System.out.println("Enter the Keyword:");
                        String key1=read.readLine();
                        System.out.println("Enter the message");
                        String msg=read.readLine();
                        String cipheTxt=cipher(msg,key1);
                        System.out.println("Ciphered As:" + cipheTxt);
                        users.get(username).cipherHistory.add("VIGENERE 
CIPHER : FROM "+msg+" TO "+cipheTxt+" USING KEY:"+key1+" ON "+d1);
                    }
                    else if(choice==2)
                    {
                        BufferedReader read=new BufferedReader(new 
InputStreamReader(System.in));
                        System.out.println("Enter the Keyword:");
                        String key1=read.readLine();
                        System.out.println("Enter the message");
                        String msg=read.readLine();
                        String cipheTxt=decipher(msg,key1);
                        System.out.println("Your message:" + cipheTxt);
                        users.get(username).deCipherHistory.add("VIGENERE 
   CIPHER : FROM "+msg+" TO "+cipheTxt+" USING KEY:"+key1+" ON "+d1);
                    }
            }
        }


   package sherlock;
   import java.util.Date;
   import java.util.HashMap;
   import java.util.Scanner;
   public class PlayFairCipher
   {
       private String KeyWord = new String();
       private String Key = new String();
       private char   matrix_arr[][] = new char[5][5];

       public void setKey(String k)
       {
           String K_adjust = new String();
           boolean flag = false;
           K_adjust = K_adjust + k.charAt(0);
           for (int i = 1; i < k.length(); i++)
           {
               for (int j = 0; j < K_adjust.length(); j++)
               {
                   if (k.charAt(i) == K_adjust.charAt(j))
                   {
                       flag = true;
                   }
               }
               if (flag == false)
                   K_adjust = K_adjust + k.charAt(i);
               flag = false;
           }
           KeyWord = K_adjust;
       }

       public void KeyGen()
       {
           boolean flag = true;
           char current;
           Key = KeyWord;
           for (int i = 0; i < 26; i++)
           {
               current = (char) (i + 97);
               if (current == 'j')
                   continue;
               for (int j = 0; j < KeyWord.length(); j++)
               {
                   if (current == KeyWord.charAt(j))
                   {
                       flag = false;
                       break;
                   }
               }
               if (flag)
                   Key = Key + current;
               flag = true;
           }
           matrix();
       }

       private void matrix()
       {
           int counter = 0;
           for (int i = 0; i < 5; i++)
           {
               for (int j = 0; j < 5; j++)
               {
                   matrix_arr[i][j] = Key.charAt(counter);
                   counter++;
               }
           }
       }

       private String format(String old_text)
       {
           int i = 0;
           int len = 0;
           String text = new String();
           len = old_text.length();
           for (int tmp = 0; tmp < len; tmp++)
           {
               if (old_text.charAt(tmp) == 'j')
               {
                   text = text + 'i';
               }
               else
                   text = text + old_text.charAt(tmp);
           }
           len = text.length();
           for (i = 0; i < len; i = i + 2)
           {
               if (text.charAt(i + 1) == text.charAt(i))
               {
                   text = text.substring(0, i + 1) + 'x' + text.substring(i
   + 1);
               }
           }
           return text;
       }

       private String[] Divid2Pairs(String new_string)
       {
           String Original = format(new_string);
           int size = Original.length();
           if (size % 2 != 0)
           {
               size++;
               Original = Original + 'x';
           }
           String x[] = new String[size / 2];
           int counter = 0;
           for (int i = 0; i < size / 2; i++)
           {
               x[i] = Original.substring(counter, counter + 2);
               counter = counter + 2;
           }
           return x;
       }

       public int[] GetDiminsions(char letter)
       {
           int[] key = new int[2];
           if (letter == 'j')
               letter = 'i';
           for (int i = 0; i < 5; i++)
           {
               for (int j = 0; j < 5; j++)
               {
                   if (matrix_arr[i][j] == letter)
                   {
                       key[0] = i;
                       key[1] = j;
                       break;
                   }
               }
           }
           return key;
       }

       public String encryptMessage(String Source)
       {
           String src_arr[] = Divid2Pairs(Source);
           String Code = new String();
           char one;
           char two;
           int part1[] = new int[2];
           int part2[] = new int[2];
           for (int i = 0; i < src_arr.length; i++)
           {
               one = src_arr[i].charAt(0);
               two = src_arr[i].charAt(1);
               part1 = GetDiminsions(one);
               part2 = GetDiminsions(two);
               if (part1[0] == part2[0])
               {
                   if (part1[1] < 4)
                       part1[1]++;
                   else
                       part1[1] = 0;
                   if (part2[1] < 4)
                       part2[1]++;
                   else
                       part2[1] = 0;
               }
               else if (part1[1] == part2[1])
               {
                   if (part1[0] < 4)
                       part1[0]++;
                   else
                       part1[0] = 0;
                   if (part2[0] < 4)
                       part2[0]++;
                   else
                       part2[0] = 0;
               }
               else
               {
                   int temp = part1[1];
                   part1[1] = part2[1];
                   part2[1] = temp;
               }
               Code = Code + matrix_arr[part1[0]][part1[1]]+
   matrix_arr[part2[0]][part2[1]];
           }
           return Code;
       }

       public String decryptMessage(String Code)
       {
           String Original = new String();
           String src_arr[] = Divid2Pairs(Code);
           char one;
           char two;
           int part1[] = new int[2];
           int part2[] = new int[2];
           for (int i = 0; i < src_arr.length; i++)
           {
               one = src_arr[i].charAt(0);
               two = src_arr[i].charAt(1);
               part1 = GetDiminsions(one);
               part2 = GetDiminsions(two);
               if (part1[0] == part2[0])
               {
                   if (part1[1] > 0)
                       part1[1]--;
                   else
                       part1[1] = 4;
                   if (part2[1] > 0)
                       part2[1]--;
                   else
                       part2[1] = 4;
               }
               else if (part1[1] == part2[1])
               {
                   if (part1[0] > 0)
                       part1[0]--;
                   else
                       part1[0] = 4;
                   if (part2[0] > 0)
                       part2[0]--;
                   else
                       part2[0] = 4;
               }
               else
               {
                   int temp = part1[1];
                   part1[1] = part2[1];
                   part2[1] = temp;
               }
               Original = Original + matrix_arr[part1[0]][part1[1]]+
   matrix_arr[part2[0]][part2[1]];
           }
           return Original;
       }

       public static void playFairCipherstart(HashMap<String,User>
   users,String username) throws Exception
       {
           PlayFairCipher x = new PlayFairCipher();
           Scanner sc = new Scanner(System.in);
           int option;
           System.out.println("1.Cipher\n2.Decipher");
           option=sc.nextInt();
           Date d1 = new Date();
           if(option==1)
           {
            System.out.println("Enter a keyword:");
            String keyword = sc.next();
            x.setKey(keyword);
            x.KeyGen();
            System.out.println("Enter word to encrypt(Message length should
   be even[if odd add z to the end]):");
            String key_input = sc.next();
            if (key_input.length() % 2 == 0)
            {
                System.out.println("Encryption:" +
   x.encryptMessage(key_input));
                users.get(username).cipherHistory.add("PLAY FAIR CIPHER :
   FROM "+key_input+" TO "+x.encryptMessage(key_input)+" USING
   KEY:"+keyword+" ON "+d1);

            }
            else
            {
                System.out.println("Message length should be even");
            }
            sc.close();
           }
           if(option==2)
           {
                System.out.println("Enter a keyword:");
               String keyword = sc.next();
               x.setKey(keyword);
               x.KeyGen();
               System.out.println("Enter word to decrypt(Message length
   should be even[if odd add z to the end]):");
               String key_input = sc.next();
               if (key_input.length() % 2 == 0)
               {
                   System.out.println("Decryption:" +
   x.decryptMessage(key_input));
                   users.get(username).deCipherHistory.add("PLAY FAIR
   CIPHER : FROM "+key_input+" TO "+x.decryptMessage(key_input)+" USING
   KEY:"+keyword+" ON "+d1);
               }
               else
               {
                   System.out.println("Message length should be even");
               }
               sc.close();
           }
       }
   }
```

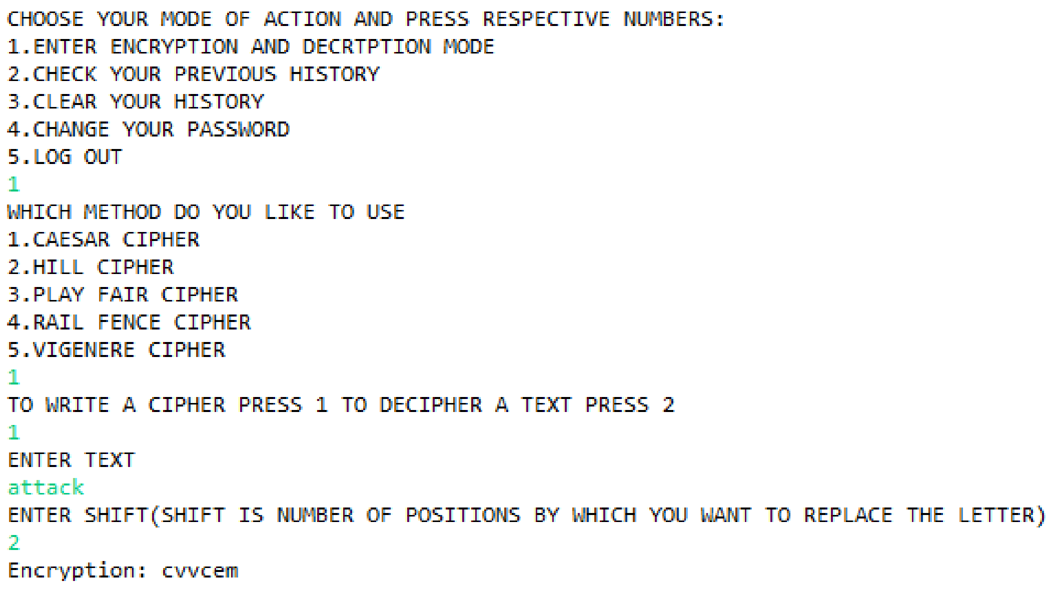
RESULTS

USER DETAILS SIGN UP AND LOGIN



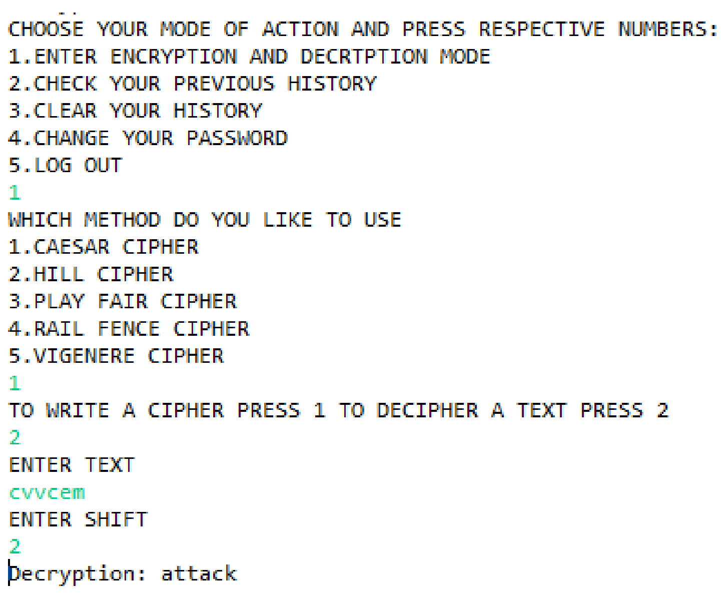
```
TO LOGIN PRESS 1 TO SIGNUP PRESS 2
1
ENTER YOUR USER NAME
gopi
ENTER YOUR PASSWORD
kittu
LOGIN SUCCESSFUL...WELCOME BACK
```

CAESAR CIPHER MODE IMPLEMENTATION



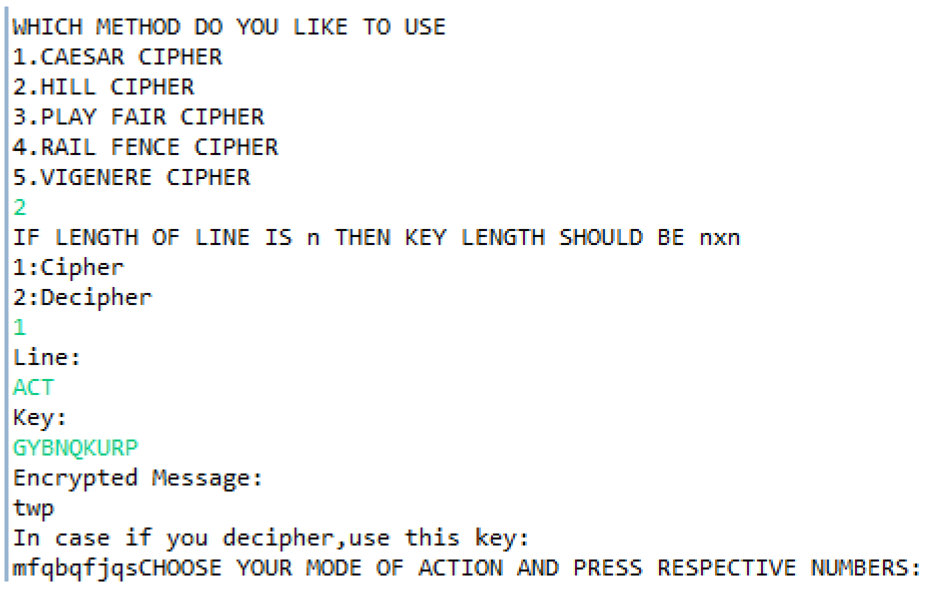
```
CHOOSE YOUR MODE OF ACTION AND PRESS RESPECTIVE NUMBERS:
1.ENTER ENCRYPTION AND DECRTPTION MODE
2.CHECK YOUR PREVIOUS HISTORY
3.CLEAR YOUR HISTORY
4.CHANGE YOUR PASSWORD
5.LOG OUT
1
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
1
TO WRITE A CIPHER PRESS 1 TO DECIPHER A TEXT PRESS 2
1
ENTER TEXT
attack
ENTER SHIFT(SHIFT IS NUMBER OF POSITIONS BY WHICH YOU WANT TO REPLACE THE LETTER)
2
Encryption: cvvcem
```

CAESAR DECIPHER MODE IMPLEMENTATION



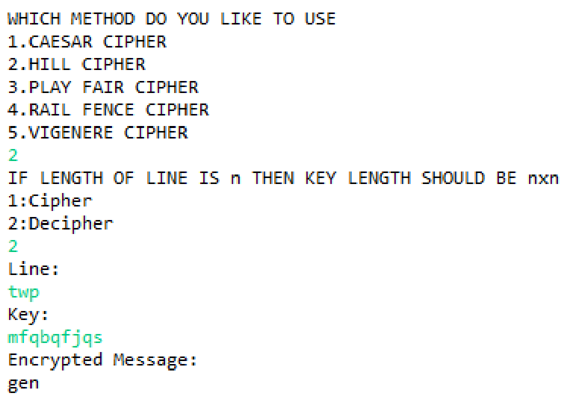
```
CHOOSE YOUR MODE OF ACTION AND PRESS RESPECTIVE NUMBERS:
1.ENTER ENCRYPTION AND DECRTPTION MODE
2.CHECK YOUR PREVIOUS HISTORY
3.CLEAR YOUR HISTORY
4.CHANGE YOUR PASSWORD
5.LOG OUT
1
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
1
TO WRITE A CIPHER PRESS 1 TO DECIPHER A TEXT PRESS 2
2
ENTER TEXT
cvvcem
ENTER SHIFT
2
Decryption: attack
```

HILL CIPHER MODE



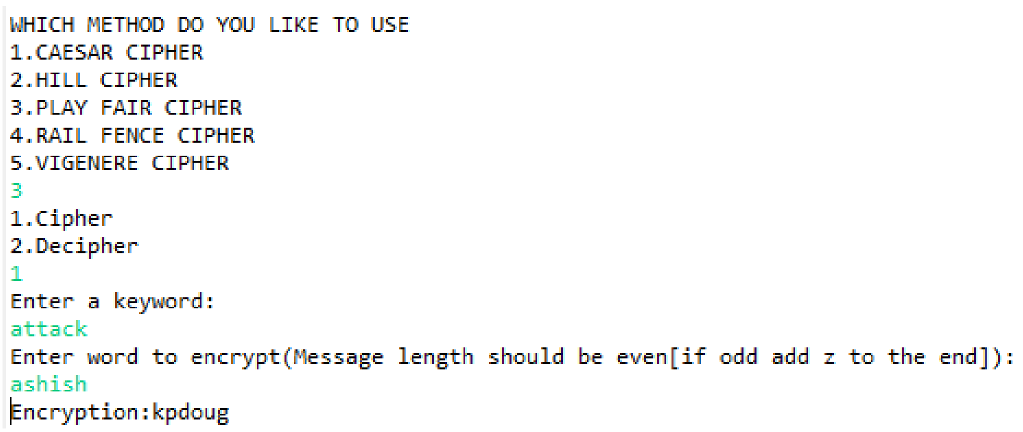
```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
2
IF LENGTH OF LINE IS n THEN KEY LENGTH SHOULD BE nxn
1:Cipher
2:Decipher
1
Line:
ACT
Key:
GYBNQKURP
Encrypted Message:
twp
In case if you decipher,use this key:
mfqbqfjqsCHOOSE YOUR MODE OF ACTION AND PRESS RESPECTIVE NUMBERS:
```

HILL DECIPHER MODE

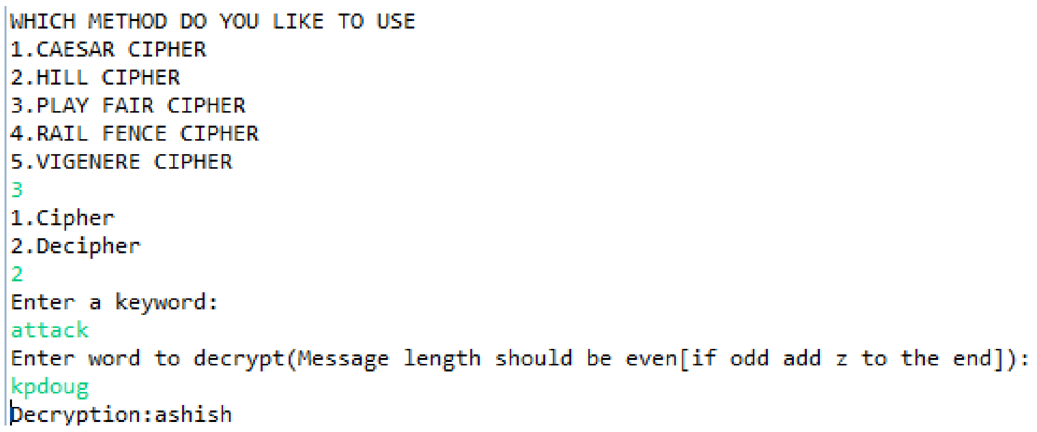


```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
2
IF LENGTH OF LINE IS n THEN KEY LENGTH SHOULD BE nxn
1:Cipher
2:Decipher
2
Line:
twp
Key:
mfqbqfjqs
Encrypted Message:
gen
```

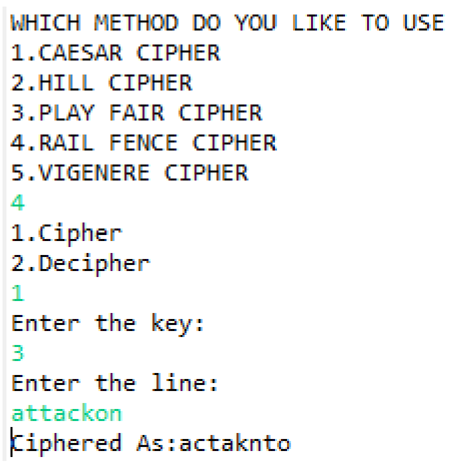
PLAYFAIR CIPHER MODE


```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
3
1.Cipher
2.Decipher
1
Enter a keyword:
attack
Enter word to encrypt(Message length should be even[if odd add z to the end]):
ashish
Encryption:kpdoug
```

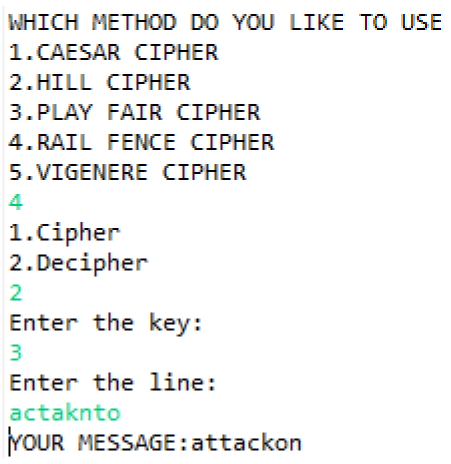
PLAYFAIR DECIPHER MODE


```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
3
1.Cipher
2.Decipher
2
Enter a keyword:
attack
Enter word to decrypt(Message length should be even[if odd add z to the end]):
kpdoug
Decryption:ashish
```

RAILFENCE CIPHER MODE

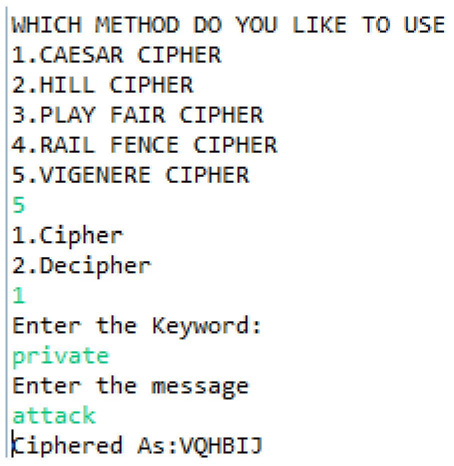

```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
4
1.Cipher
2.Decipher
1
Enter the key:
3
Enter the line:
attackon
Ciphered As:actaknto
```

RAILFENCE DECIPHER MODE



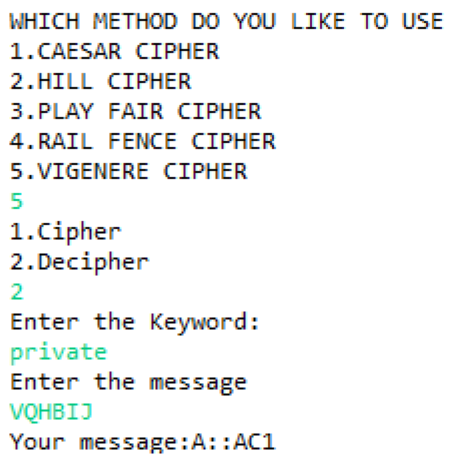
```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
4
1.Cipher
2.Decipher
2
Enter the key:
3
Enter the line:
actaknto
YOUR MESSAGE:attackon
```

VIGNERE CIPHER MODE



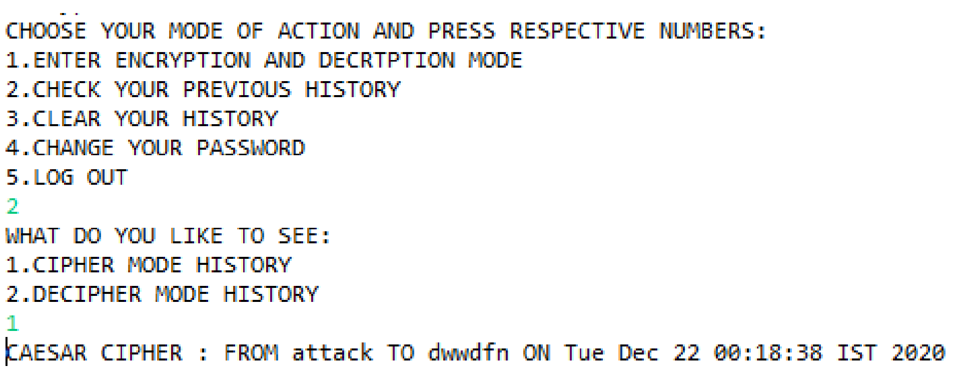
```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
5
1.Cipher
2.Decipher
1
Enter the Keyword:
private
Enter the message
attack
Ciphered As:VQHBIJ
```

VIGENERE DECIPHER MODE



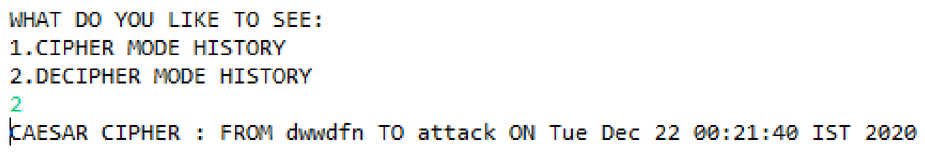
```
WHICH METHOD DO YOU LIKE TO USE
1.CAESAR CIPHER
2.HILL CIPHER
3.PLAY FAIR CIPHER
4.RAIL FENCE CIPHER
5.VIGENERE CIPHER
5
1.Cipher
2.Decipher
2
Enter the Keyword:
private
Enter the message
VQHBIJ
Your message:A::AC1
```

THE USER CIPHER HISTORY IS STORED IN FILE AS SHOWN AS OUTPUT BELOW



```
CHOOSE YOUR MODE OF ACTION AND PRESS RESPECTIVE NUMBERS:
1.ENTER ENCRYPTION AND DECRTPTION MODE
2.CHECK YOUR PREVIOUS HISTORY
3.CLEAR YOUR HISTORY
4.CHANGE YOUR PASSWORD
5.LOG OUT
2
WHAT DO YOU LIKE TO SEE:
1.CIPHER MODE HISTORY
2.DECIPHER MODE HISTORY
1
CAESAR CIPHER : FROM attack TO dwwdfn ON Tue Dec 22 00:18:38 IST 2020
```

THE USER DECIPHER HISTORY IS SHOWN



```
WHAT DO YOU LIKE TO SEE:
1.CIPHER MODE HISTORY
2.DECIPHER MODE HISTORY
2
CAESAR CIPHER : FROM dwwdfn TO attack ON Tue Dec 22 00:21:40 IST 2020
```

CONCLUSION

In conclusion, our application provides high privacy for the users accessing and check them whenever they need to access it. The users can decrypt and crack the codes which would be not be able to do with normal intelligence.

---THE END-------------------------------------