Dharmsinh Desai University, Nadiad
Department of Information Technology
DAIE, IT704
B.Tech. IT, Sem: VII

Submitted By

Roll No: IT076
Name: Dishant Modh

Experiment 3: Write a C/C++/Java program to implement Hill cipher.

1. Code

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class HillCipher
{
    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);
   }
   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))
```

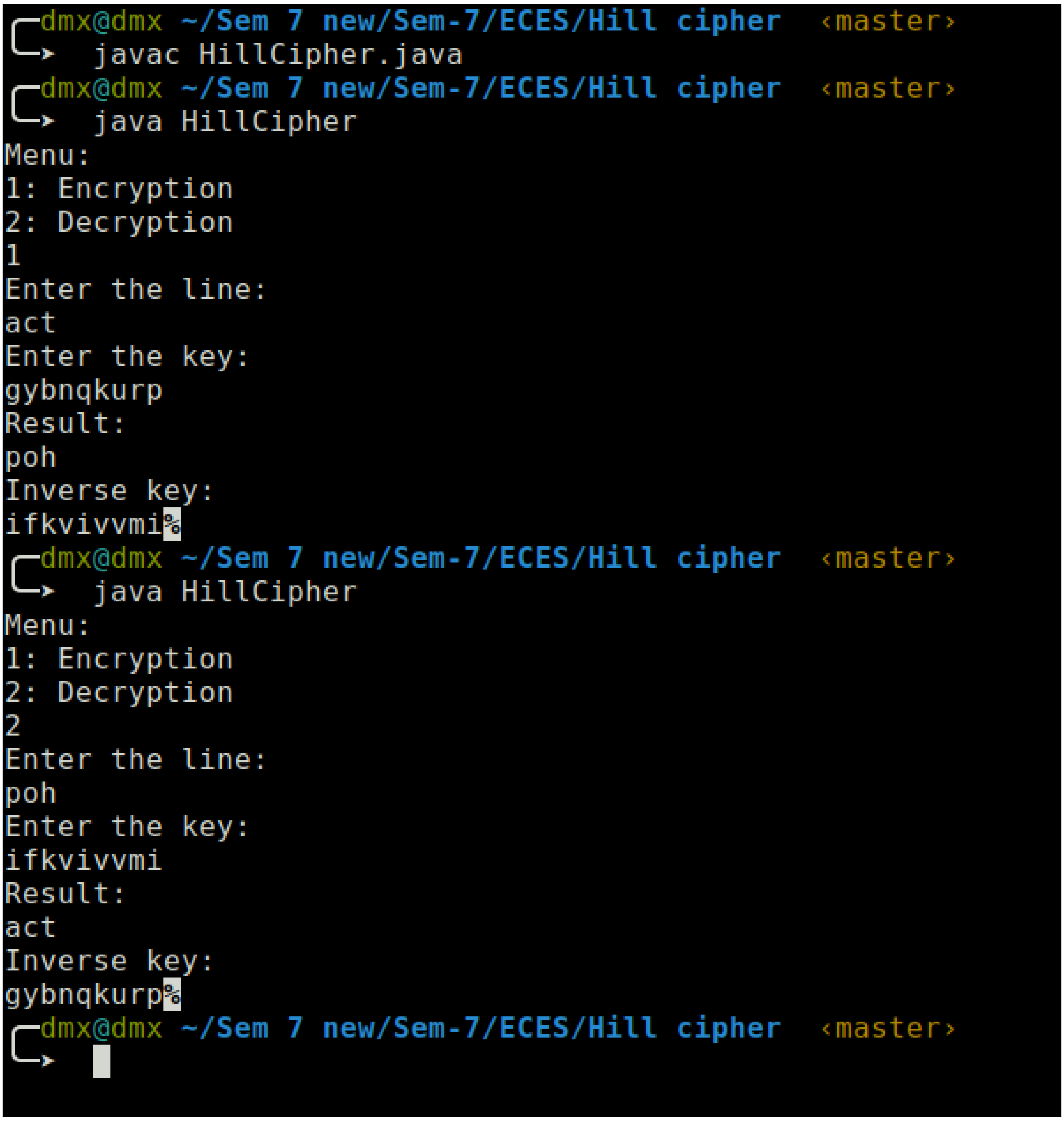
```
                            n++;
                        else
                        {
                            n = 0;
                            m++;
                        }
                    }
                }
            }
            fac[q][p] = (int) Math.pow(-1, q + p) * determinant(b, f - 1);
        }
    }
    trans(fac, f);
}
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("\nInverse 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 main(String args[]) throws IOException

    {

        HillCipher obj = new HillCipher();

        BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

        int choice;

        System.out.println("Menu:\n1: Encryption\n2: Decryption");

        choice = Integer.parseInt(in.readLine());

        System.out.println("Enter the line: ");

        String line = in.readLine();

        System.out.println("Enter the key: ");
        String key = in.readLine();

        double sq = Math.sqrt(key.length());

        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("Result:");

                obj.divide(line, s);

                obj.cofact(obj.keymatrix, s);

            }

        }

    }

}
```

2. Output



```
╭─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Hill cipher  ‹master›
╰─➤  javac HillCipher.java
╭─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Hill cipher  ‹master›
╰─➤  java HillCipher
Menu:
1: Encryption
2: Decryption
1
Enter the line:
act
Enter the key:
gybnqkurp
Result:
poh
Inverse key:
ifkvivvmi%
╭─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Hill cipher  ‹master›
╰─➤  java HillCipher
Menu:
1: Encryption
2: Decryption
2
Enter the line:
poh
Enter the key:
ifkvivvmi
Result:
act
Inverse key:
gybnqkurp%
╭─dmx@dmx ~/Sem 7 new/Sem-7/ECES/Hill cipher  ‹master›
╰─➤  
```