Welcome to cryptographic encryption technique:

1.Caesar cipher

2.Monoalphabetic cipher

3.Homophonic substitution cipher

4.Polygram substitution cipher

5.Vigenere cipher

------ENTER YOUR CHOICE OF ENCRYPTION------

1

Enter the cipher point : 5

Enter a message to encrypt:Sample

Encrypted message: Xfruqj

Welcome to cryptographic encryption technique:

1.Caesar cipher

2.Monoalphabetic cipher

3.Homophonic substitution cipher

4.Polygram substitution cipher

5.Vigenere cipher

------ENTER YOUR CHOICE OF ENCRYPTION------

2

s[0]:q

s[1]:w

s[2]:e

s[3]:r

s[4]:t

s[5]:y

s[6]:u

s[7]:i

s[8]:o

s[9]:p

s[10]:a

s[11]:s

s[12]:d

s[13]:f

s[14]:g

s[15]:h

s[16]:j

s[17]:k

s[18]:l

s[19]:z

s[20]:x

s[21]:c

s[22]:v

s[23]:b

s[24]:n

s[25]:m

Enter your statement:

sample

lqdhst

Welcome to cryptographic encryption technique:

1.Caesar cipher

2.Monoalphabetic cipher

3.Homophonic substitution cipher

4.Polygram substitution cipher

5.Vigenere cipher

------ENTER YOUR CHOICE OF ENCRYPTION------

3

Enter a message to encrypt: sample

102109260722

Welcome to cryptographic encryption technique:

1.Caesar cipher

2.Monoalphabetic cipher

3.Homophonic substitution cipher

4.Polygram substitution cipher

5.Vigenere cipher

------ENTER YOUR CHOICE OF ENCRYPTION------

4

Enter the total no of letters you want to shift(multiple of 3):

9

Enter the plaintext :

sample

Your Text is:

sample

Your plaintext matrix is:

18 0 12 32759 11 4 0 0 -476977248

The Encrypted matrix is:

4 24 22 6 8 18 -20 -14 -8

The Encrypted text is:

EYWGIS

Welcome to cryptographic encryption technique:

1.Caesar cipher

2.Monoalphabetic cipher

3.Homophonic substitution cipher

4.Polygram substitution cipher

5.Vigenere cipher

------ENTER YOUR CHOICE OF ENCRYPTION------

5

Enter the plain text:sample

Enter the key:5

New key is:555555

Cipher text is: aO[^ZS

#include "bits/stdc++.h"

using namespace std;

// Class for defining a Tree Node;

class Node{

   public:

   int value;

   Node\* left;

   Node\* right;

};

// Function to construct a new Node;

Node\* NewNode(int val){

   Node\* node = new Node;

   node->value = val;

   node->left = NULL;

   node->right = NULL;

   return node;

}

//To check if a leaf node;

bool leaf(Node\* root){

   return(!root->left && !root->right);

}

//Function which constructs the Sum Tree;

int SumTree(Node\* root){

   //If leaf node, the we simply put 0 in that node and returns

   if(leaf(root)){

       int old\_ = root->value;

       //Storing value of 0 in leaf node;

       root->value = 0;

       //returns the old value;

       return old\_ + root->value;

   }

   //Storing the old value;

   int old\_ = root->value;

   //Storing the value of sum of left + right subtree;

   root->value = SumTree(root->left) + SumTree(root->right);