**IMPLEMENTATION OF CAESAR CIPHER**

**AIM:**

To write a program to implement caesar cipher which can handle function like:

1. Returning a ciphertext for a given plain text with key

2. Returning the plain text for the given ciphertext with a key

3. Returning all possible combinations of plain text for a given cipher text

**ALGORITHM:**

1. START

2. Print the 3 functions that are provided to the user.

a. Returning a ciphertext for a given plain text with key

b. Returning the plain text for the given ciphertext with a key

c. Returning all possible combinations of plain text for a given cipher text

d. Quit

3.If choice is 1,get the message input from the user with key. For each word , add its ASCII value with key and take the modulus with 26. It would remove the first value, so to convert back to character we add ‘a’ to the result

4. Add the characters back to a string and print the encrypted result.

5. If the choice is 2, get the cipher text with the key from user. Follow the step 3 as in encryption but subtract the key and print the result.

6. If choice is 3, get text from user and run a loop from 1 to 25. Follow the step 3 and print the result in each iteration.

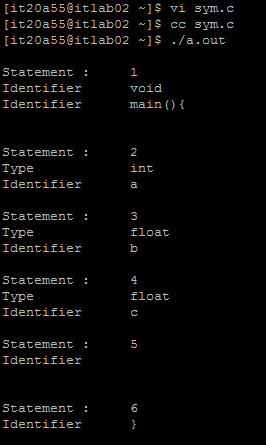
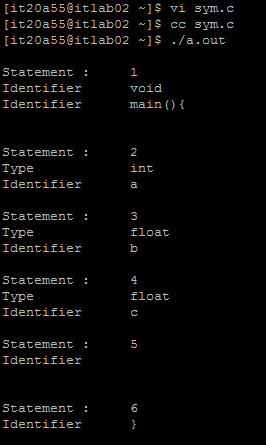
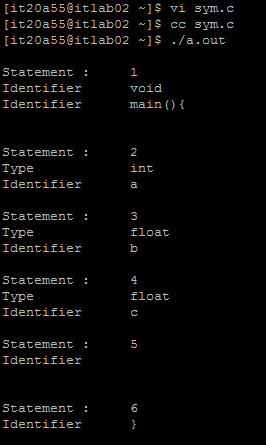
7. End if choice is 4.

8. END

**CODE:**

|  |
| --- |
| #include<stdio.h>  #include<ctype.h>  void encrypt(){  char text[500], ch;  int key;  printf("Enter the message to encrypt:\n");  scanf("%s",text);  printf("enter the key:");  scanf("%d",&key);  for (int i = 0; text[i] != '\0'; ++i) {  ch = text[i];  if (isalnum(ch)) {  if (islower(ch)) {  ch = (ch - 'a' + key) % 26 + 'a';  }  if (isupper(ch)) {  ch = (ch - 'A' + key) % 26 + 'A';  }  if (isdigit(ch)) {  ch = (ch - '0' + key) % 10 + '0';  }  }  else {  printf("Invalid Message");  }  text[i] = ch;  }  printf("Encrypted message: %s\n", text);  }  void decrypt(){  char text[500], ch;  int key;  printf("Enter the message to decrypt:");  scanf("%s",text);  printf("enter the key:");  scanf("%d",&key);  for (int i = 0; text[i] != '\0'; ++i) {  ch = text[i];  if (isalnum(ch)) {  if (islower(ch)) {  ch = (ch - 'a' - key) % 26 + 'a';  }  if (isupper(ch)) {  ch = (ch - 'A' - key) % 26 + 'A';  }  if (isdigit(ch)) {  ch = (ch - '0' - key) % 10 + '0';  }  }  else {  printf("Invalid Message");  }  text[i] = ch;  }  printf("Decrypted message:%s",text);  }  void looping(){  char text[500],ch;  int key;  printf("Enter the message:");  scanf("%s",text);  for(key=1;key<26;key++){  for (int i = 0; text[i] != '\0'; ++i) {  ch = text[i];  if (isalnum(ch)) {  if (islower(ch)) {  ch = (ch - 'a' + key) % 26 + 'a';  }  if (isupper(ch)) {  ch = (ch - 'A' + key) % 26 + 'A';  }  if (isdigit(ch)) {  ch = (ch - '0' + key) % 10 + '0';  }  }  else {  printf("Invalid Message");  }  text[i] = ch;  }  printf("Encrpted message:%s\n",text);  }  }  int main()  {  int n=0;  int s;  bool flag=true;  while(flag){  printf("Enter the choice:");  scanf("%d",&n);  if(n==1){  encrypt();  }  else if(n==2){  decrypt();  }else if(n==3){  looping();  }  else{  printf("Invalid choice");  }  printf("\nDo you want to continue?0/1");  scanf("%d",&s);  if(s==1){  flag=false;  }  else{  flag=true;  }  }  return 0;  } |

**OUTPUT:**

****

**RESULT:**

Hence the Caesar cipher has been implemented successfully in c program.

**IMPLEMENTATION OF PLAYFAIR CIPHER**

**AIM:**

To implement playfair cipher with functions like:

1. Display the playfair matrix for the given key

2. Return the cipher text for the given plain text

3. Return the plain text for the given cipher text

**ALGORITHM:**

1. Start

2. Print the function available to get the choice from the user:

a. Display the playfair matrix for the given key

b. Return the cipher text for the given plain text

c. Return the plain text for the given cipher text

d. Quit

3. Get the choice from the user

4. If the choice is ‘a’, then construct the 5 x 5 matrix by getting a key. Iterate over the given

key and if the character is not present in the matrix, then add it to the matrix. After using all

the characters in the matrix, iterate from a to z and if the character is not present in the

matrix, then add it to the matrix. Finally print the matrix to the user

5. If the choice is ‘b’, then get the plain text and the key from the user. Construct the matrix

as said in step 4. Pair the words in the plain text in a way that 2 alphabets should not be the

same while pairing. If they are the same, then add ‘x’ between them. For each pair, look on

to the matrix.

5.1 If they are in the same row, then the right character should be their cipher

character

5.2 If they are in the same column, then the character down should be their cipher

character

5.3 If they are in different rows and columns, then the character which is

intersecting the others column in the same row should be their cipher character.

Print the cipher text to the user.

6. If the choice is ‘c’, then get the cipher text and the key. Build the matrix as said in step 4.

Perform the same operations as said in the step 5 but in the opposite direction for the pairs

in the same row or same column. Print the plain text to the user

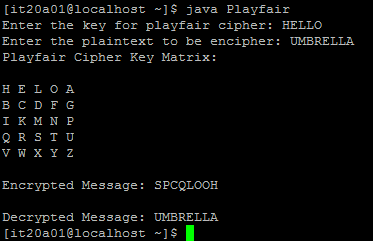
7. If the choice is ‘d’, then quit

8. End

**CODE:**

|  |
| --- |
| import java.awt.Point;   import java.util.Scanner;   public class Playfair {    private int length = 0;    private String [][] table;   public static void main(String args[])   {   Playfairr pf = new Playfair ();   }    private PlayfairCipher()   {    System.out.print("Enter the key for playfair cipher: ");   Scanner sc = new Scanner(System.in);   String key = parseString(sc);   while(key.equals(""))   key = parseString(sc);   table = this.cipherTable(key);    System.out.print("Enter the plaintext to be encipher: ");    String input = parseString(sc);   while(input.equals(""))   input = parseString(sc);   String output = cipher(input);   String decodedOutput = decode(output);   this.keyTable(table);   this.printResults(output,decodedOutput);   }    private String parseString(Scanner sc)   {   String parse = sc.nextLine();   parse = parse.toUpperCase();   parse = parse.replaceAll("[^A-Z]", "");   parse = parse.replace("J", "I");   return parse;   }     private String[][] cipherTable(String key)   {   String[][] playfairTable = new String[5][5];   String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";   for(int i = 0; i < 5; i++)   for(int j = 0; j < 5; j++)   playfairTable[i][j] = "";   for(int k = 0; k < keyString.length(); k++)   {   boolean repeat = false;   boolean used = false;   for(int i = 0; i < 5; i++)   {   for(int j = 0; j < 5; j++)   {   if(playfairTable[i][j].equals("" + keyString.charAt(k)))   {   repeat = true;   }   else if(playfairTable[i][j].equals("") && !repeat && !used)   {   playfairTable[i][j] = "" + keyString.charAt(k);   used = true;   }   }   }   }   return playfairTable;   }   private String cipher(String in)   {   length = (int) in.length() / 2 + in.length() % 2;   for(int i = 0; i < (length - 1); i++)   {   if(in.charAt(2 \* i) == in.charAt(2 \* i + 1))   {   in = new StringBuffer(in).insert(2 \* i + 1, 'X').toString();   length = (int) in.length() / 2 + in.length() % 2;   }   }   String[] digraph = new String[length];   for(int j = 0; j < length ; j++)   {   if(j == (length - 1) && in.length() / 2 == (length - 1))   in = in + "X";   digraph[j] = in.charAt(2 \* j) +""+ in.charAt(2 \* j + 1);   }    String out = "";   String[] encDigraphs = new String[length];   encDigraphs = encodeDigraph(digraph);   for(int k = 0; k < length; k++)   out = out + encDigraphs[k];   return out;   }   private String[] encodeDigraph(String di[])   {   String[] encipher = new String[length];   for(int i = 0; i < length; i++)   {   char a = di[i].charAt(0);   char b = di[i].charAt(1);   int r1 = (int) getPoint(a).getX();   int r2 = (int) getPoint(b).getX();   int c1 = (int) getPoint(a).getY();   int c2 = (int) getPoint(b).getY();   if(r1 == r2)   {   c1 = (c1 + 1) % 5;   c2 = (c2 + 1) % 5;   }    else if(c1 == c2)   {   r1 = (r1 + 1) % 5;   r2 = (r2 + 1) % 5;   }    else   {   int temp = c1;   c1 = c2;   c2 = temp;   }    encipher[i] = table[r1][c1] + "" + table[r2][c2];   }   return encipher;   }    private String decode(String out)   {   String decoded = "";   for(int i = 0; i < out.length() / 2; i++)   {   char a = out.charAt(2\*i);   char b = out.charAt(2\*i+1);   int r1 = (int) getPoint(a).getX();   int r2 = (int) getPoint(b).getX();   int c1 = (int) getPoint(a).getY();   int c2 = (int) getPoint(b).getY();   if(r1 == r2)   {   c1 = (c1 + 4) % 5;   c2 = (c2 + 4) % 5;   }   else if(c1 == c2)   {   r1 = (r1 + 4) % 5;   r2 = (r2 + 4) % 5;   }   else   {       int temp = c1;   c1 = c2;   c2 = temp;   }   decoded = decoded + table[r1][c1] + table[r2][c2];   }     return decoded;   }    private Point getPoint(char c)   {   Point pt = new Point(0,0);   for(int i = 0; i < 5; i++)   for(int j = 0; j < 5; j++)   if(c == table[i][j].charAt(0))   pt = new Point(i,j);   return pt;   }   private void keyTable(String[][] printTable)   {   System.out.println("Playfair Cipher Key Matrix: ");   System.out.println();    for(int i = 0; i < 5; i++)   {    for(int j = 0; j < 5; j++)   {    System.out.print(printTable[i][j]+" ");   }   System.out.println();   }   System.out.println();   }       private void printResults(String encipher, String dec)   {   System.out.print("Encrypted Message: ");     System.out.println(encipher);   System.out.println();   System.out.print("Decrypted Message: ");    System.out.println(dec);   }   } |

**OUTPUT:**

****

**RESULT:**

Hence the play fair cipher has been successfully implemented in java programming language.

**IMPLEMENTATION OF VIGNERE CIPHER**

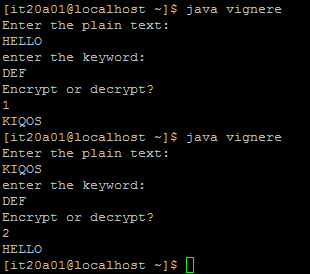
**AIM:**

**ALGORITHM:**

**CODE:**

|  |
| --- |
| import java.util.\*;  class Main{  static String generatekey(String str,String key){  int x=str.length();  for(int i=0;i>=0;i++){  if(x==i){  i=0;  }  if(key.length()==str.length()){  break;  }  key+=(key.charAt(i));  }  return key;  }  static String encrypt(String str,String key){  String ctext="";  for(int i=0;i<key.length() && i<str.length();i++){  int ch = (str.charAt(i) + key.charAt(i))%26;  ch+='A';  ctext += (char)(ch);  }  return ctext;  }    static String decrypt(String str,String key){  String ctext="";  for(int i=0;i<key.length() && i<str.length();i++){  int ch = (str.charAt(i) - key.charAt(i))%26;  ch+='A';  ctext += (char)(ch);  }  return ctext;  }  public static void main(String args[]){  Scanner scan = new Scanner(System.in);  System.out.println("Enter the plain text:");  String text = scan.nextLine();  System.out.println("enter the keyword:");  String key=scan.nextLine();  String keyword=generatekey(text,key);  System.out.println("Encrypt or decrypt?");  int ch=scan.nextInt();  if(ch==1){  String cipher=encrypt(text,keyword);  System.out.println(cipher);  }  else if(ch==2){  String original=decrypt(text,keyword);  System.out.println(original);  }  else{  System.out.println("Invalid");  }  }  } |

**OUTPUT:**

****

**RESULT:**

Hence the vignere cipher has been successfully implemented.