**PRACTICAL-5**

**Aim: Prepare encryption of plain text and decryption of ciphertext using additive cipher (CAESAR CIPHER).**

**Theory:**

**Example for encryption**

Use the additive cipher with key = 15 to encrypt the message “hello”.

**Solution**

We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h → 07 encryption: (07 + 15) mod 26 cirphertext: 22 → W

Plaintext: e → 04 encryption: (04 + 15) mod 26 cirphertext: 19 → T

Plaintext: l → 11 encryption: (11 + 15) mod 26 cirphertext: 00 → A

Plaintext: l → 11 encryption: (11 + 15) mod 26 cirphertext: 00 → A

Plaintext: o → 14 encryption: (14 + 15) mod 26 cirphertext: 03 → D

**Table5.1: convert plaintext to ciphertext**

The result is “WTAAD”. Note that the cipher is monoalphabetic because two instances of the same plaintext character (l’s) are encrypted as the same character (A).

**Example for decryption**

Use the additive cipher with key = 15 to decrypt the message “WTAAD”.

**Solution**

We apply the decryption algorithm to the plaintext character by character:

Plaintext: W → 22 encryption: (22 + 15) mod 26 cirphertext: 07 → h

Plaintext: T → 19 encryption: (19 + 15) mod 26 cirphertext: 04 → e

Plaintext: A → 00 encryption: (00 + 15) mod 26 cirphertext: 11 → l

Plaintext: A → 00 encryption: (00 + 15) mod 26 cirphertext: 11 → l

Plaintext: D → 03 encryption: (03 + 15) mod 26 cirphertext: 14 → o

**Table5.2: convert ciphertext to plaintext (using same key)**

To decrypt the message “WTAAD”. Use the additive inverse of key 15 is 11.

Plaintext: W → 22 encryption: (22 + 11) mod 26 cirphertext: 07 → h

Plaintext: T → 19 encryption: (19 + 11) mod 26 cirphertext: 04 → e

Plaintext: A → 00 encryption: (00 + 11) mod 26 cirphertext: 11 → l

Plaintext: A → 00 encryption: (00 + 11) mod 26 cirphertext: 11 → l

Plaintext: D → 03 encryption: (03 + 11) mod 26 cirphertext: 14 → o

**Table5.3: convert ciphertext to plaintext (using additive inverse of key)**

The result is “hello”. Note that the operation is in modulo 26 (see Chapter 2), which means that a negative result needs to be mapped to Z (for example − 15 becomes 11).

**Shift Cipher:**

Historically, additive ciphers are called shift ciphers. The reason is that the encryption algorithm can be interpreted as “shift key characters down” and the encryption algorithm can be interpreted as “shift key character up”. For example, if the key = 15, the encryption algorithm shifts 15 characters down (toward the end of the alphabet). The decryption algorithm shifts 15 characters up (toward the beginning of the alphabet). Of course, when we reach the end or the beginning of the alphabet, we wrap around (manifestation of modulo 26).

**Caesar Cipher**

Julius Caesar used an additive cipher to communicate with his officers. For this reason, additive ciphers are sometimes referred to as the Caesar cipher. Caesar used a key of 3 for his communications. Additive ciphers are sometimes referred to as shift ciphers or Caesar cipher.

**Program:**

**#include<iostream>**

**#include<string>**

**using namespace std;**

**void encode(char \*encode\_msg,int KEY)**

**{**

**int i;**

**for(i=0;encode\_msg[i]!='\0';i++)**

**if(encode\_msg[i]>='a' && encode\_msg[i]<='z')**

**encode\_msg[i]=(encode\_msg[i]-'a'+KEY)%26+'a';**

**}**

**void decode(char \*decode\_msg,int KEY)**

**{**

**int i;**

**for(i=0;decode\_msg[i]!='\0';i++)**

**if(decode\_msg[i]>='a' && decode\_msg[i]<='z')**

**decode\_msg[i]=(decode\_msg[i]-'a'+KEY)%26+'a';**

**}**

**void bruteforce(char \*msg)**

**{**

**char temp[100];**

**for(int i=0;i<25;i++) {**

**for(int j=0; msg[j] != '\0'; j++)**

**temp[j] = msg[j];**

**decode(temp,i+1);**

**cout << "decode message for key " << i+1 << " is : " << temp << endl;**

**}**

**}**

**int main()**

**{**

**char msg[200],code\_msg[200];**

**int key;**

**cout << "Enter message : ";**

**cin >> msg;**

**cout << "Enter Key : ";**

**cin >> key;**

**encode(msg,key);**

**cout << "Encoded Message Is : " << msg << endl;**

**cout << "\nEnter Message to Decoded: " ;**

**cin >> code\_msg;**

**bruteforce(code\_msg);**

**return 0;**

**}**

**Output:**

Enter message : todayisholiday

Enter Key : 20

Encoded Message Is : nixuscmbifcxus

Enter Message to Decoded: nixuscmbifcxus

decode message for key 1 is : ojyvtdncjgdyvt

decode message for key 2 is : pkzwueodkhezwu

decode message for key 3 is : qlaxvfpelifaxv

decode message for key 4 is : rmbywgqfmjgbyw

decode message for key 5 is : snczxhrgnkhczx

decode message for key 6 is : todayisholiday

decode message for key 7 is : upebzjtipmjebz

decode message for key 8 is : vqfcakujqnkfca

decode message for key 9 is : wrgdblvkrolgdb

decode message for key 10 is : xshecmwlspmhec

decode message for key 11 is : ytifdnxmtqnifd

decode message for key 12 is : zujgeoynurojge

decode message for key 13 is : avkhfpzovspkhf

decode message for key 14 is : bwligqapwtqlig

decode message for key 15 is : cxmjhrbqxurmjh

decode message for key 16 is : dynkiscryvsnki

decode message for key 17 is : ezoljtdszwtolj

decode message for key 18 is : fapmkuetaxupmk

decode message for key 19 is : gbqnlvfubyvqnl

decode message for key 20 is : hcromwgvczwrom

decode message for key 21 is : idspnxhwdaxspn

decode message for key 22 is : jetqoyixebytqo

decode message for key 23 is : kfurpzjyfczurp

decode message for key 24 is : lgvsqakzgdavsq

decode message for key 25 is : mhwtrblahebwtr

**Conclusion:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_