Cryptography lab day-1

Program 1:

Write a program to encrypt and decrypt a message using **Caesar cipher** along with the key

Aim: To write a c program to encrypt and decrypt a message using Caesar cipher along with the key and execute.

Algorithm:

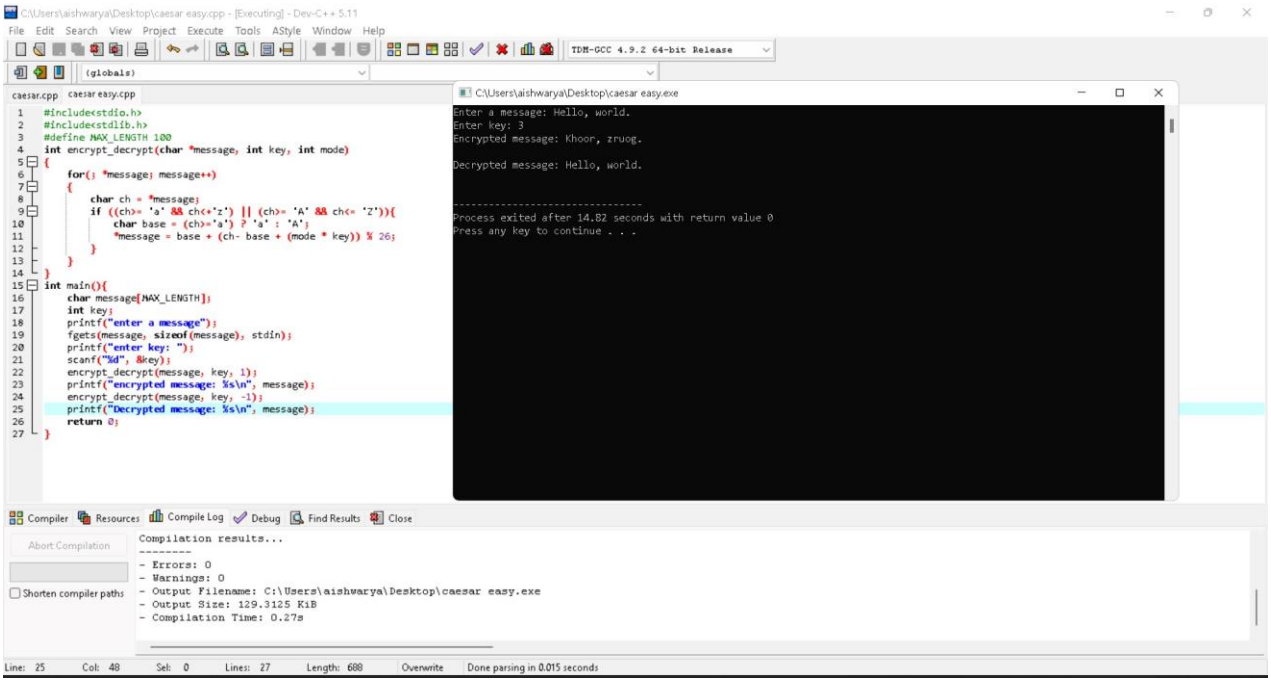
1. Take the message and the key (an integer) as input.
2. Declare a character array to store the message and read the message from the user and store it in the array.
3. Iterate through each character in the message.
4. For each alphabetic character (a-z or A-Z), shift it by the key value.
5. Print the encrypted message.
6. Take the encrypted message and the key as input.
7. Declare a character array to store the decrypted message and read the encrypted message from the user and store it in the array.
8. Iterate through each character in the encrypted message.
9. For each alphabetic character (a-z or A-Z), shift it back by the key value.
10. Print the decrypted message.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX_LENGTH 100

// Function to encrypt or decrypt the message

int encrypt_decrypt(char *message, int key, int mode) {
   for(; *message; message++) {
      char ch = *message;
      if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
         { char base = (ch >= 'a') ? 'a' : 'A';
         *message = base + (ch - base + (mode * key)) %
      26; }
   }
}
```

```
int main() { char
    message[MAX_LENGTH];
    int key;

    // Input the message

    printf("Enter a message: ");
    fgets(message, sizeof(message),
    stdin); // Input the key

    printf("Enter key:

    "); scanf("%d",

    &key); // Encrypt

    the message

    encrypt_decrypt(message, key, 1);

    printf("Encrypted message: %s\n",

    message); // Decrypt the message

    encrypt_decrypt(message, key, -1);
    printf("Decrypted message: %s\n", message);

    return 0;
}
```

Input and Output:



Result: A C program to encrypt and decrypt a message using Caesar cipher along with the key is executed successfully.

Program 2:

Perform encryption and decryption of a message using Vigenère Cipher substitution technique.

Aim: To write a c program to perform encryption and decryption of a message using Vigenere Cipher substitution technique.

Algorithm:

1. Include necessary header files (<stdio.h> and <string.h>).
2. Declare character arrays for the original message (msg), encryption key (key), new key (newKey), encrypted message (encryptedMsg), and decrypted message (decryptedMsg).
3. Declare integer variables msgLen, keyLen, i, and j for storing lengths and loop indices.
4. Initialize msg and key with the original message and encryption key.
5. Calculate msgLen and keyLen using strlen.
6. Use a loop to generate the new key (newKey) based on the original key (key).
7. Initialize i and j to 0.
8. Use a loop to iterate over each character in the original message (msg).
9. Combine it with the corresponding character from the new key (newKey) using modular arithmetic.
10. Add a null terminator at the end of the decrypted message.

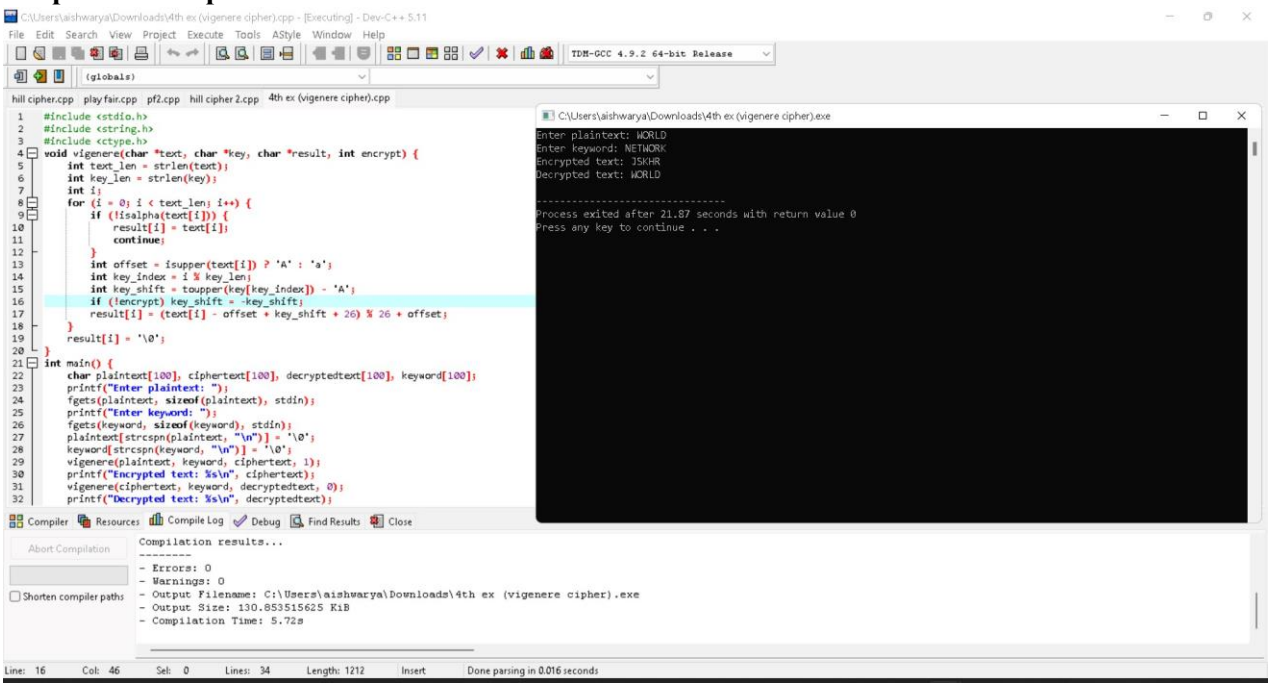
Program:

```
#include <stdio.h>

#include <string.h> #include <ctype.h> void vigenere(char *text,
char *key, char *result, int encrypt) { int text_len = strlen(text);
int key_len = strlen(key); int i;

    for (i = 0; i < text_len; i++) { if (!isalpha(text[i])) { result[i] =
        text[i]; continue; } int offset = isupper(text[i]) ? 'A' : 'a'; int
        key_index = i % key_len; int key_shift =
        toupper(key[key_index]) - 'A'; if (!encrypt) key_shift = -
        key_shift; result[i] = (text[i] - offset + key_shift + 26) % 26 +
        offset;

    }

    result[i] = '\0';
```

```
} int
main() {
char
plaintext[
100],
ciphertext
[100],
decrypted
text[100],
keyword[1
00];
printf("Ent
er
plaintext:
");
fgets(plain
text,
sizeof(plai
ntext),
stdin);
printf("Ent
er
keyword:
");
fgets(key
word,
sizeof(key
word),
stdin);
plaintext[s
trcspn(plai
ntext,
"\n")] =
'\0';
keyword[s
trcspn(key
word,
"\n")] =
'\0';
vigenere(
plaintext,
keyword,
ciphertext
, 1);
printf("En
crypted
text:
%s\n",
```

```
ciphertext
);
vigenere(c
iphertext,
keyword,
decrypted
text, 0);
printf("De
crypted
text:
%s\n",
decrypted
text);
return 0;

}
```

Input and Output:



Result: A c program to perform encryption and decryption of a message using Vigenere Cipher substitution technique is successfully executed.

Program 3:

Write a program for demonstration of encrypting and decrypting the messages by Playfair Substitution technique.

Aim: To write a c program to perform encryption and decryption of a message using Playfair substitution technique.

Algorithm:

1. Include necessary header files (<stdio.h> and <string.h>).
2. Declare character arrays for the original message (msg), encryption key (key), new key (newKey), encrypted message (encryptedMsg), and decrypted message (decryptedMsg).
3. Declare integer variables msgLen, keyLen, i, and j for storing lengths and loop indices.
4. Initialize msg and key with the original message and encryption key.
5. Calculate msgLen and keyLen using strlen.
6. Use a loop to generate the new key (newKey) based on the original key (key).
7. Initialize i and j to 0.
8. Use a loop to iterate over each character in the original message (msg).
9. Combine it with the corresponding character from the new key (newKey) using modular arithmetic.
10. Add a null terminator at the end of the decrypted message.

Program:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h> #define

SIZE 5 char

playfair[SIZE][SIZE];


void preparePlayfairKey(char *key) {

    char *ptr, *temp; int i, j,
    k, l, flag = 0; char
    alphabet[26] = {0};

    ptr = key;
    temp = key;
```

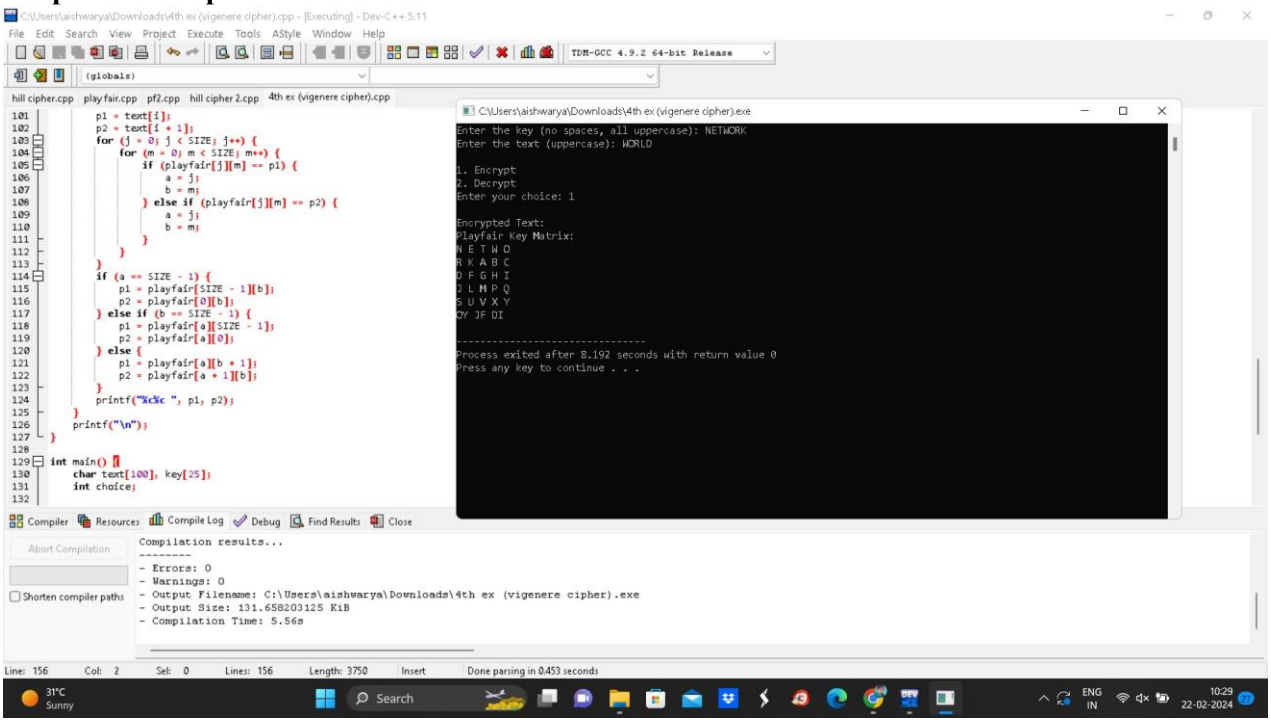
```
    while (*ptr != '\0') { if (*ptr >= 'a'
        && *ptr <= 'z') { *ptr = *ptr -
        32;

        }
        ptr++
        ;

    }

    while (*temp != '\0') {

        if (*temp == 'J') {
        *temp = 'I'; }

        if (alphabet[*temp - 65] == 0) {

            alphabet[*temp - 65] = 1; playfair[flag /
            SIZE][flag % SIZE] = *temp;

            flag+
        +; }
        temp++;

    }

    // Fill the remaining characters
    for (i = 0; i < 26; i++) {

        if (alphabet[i] == 0) {

                    playfair[flag / SIZE][flag % SIZE] = (char) (i + 65);
            flag++;

        }

    }

}

void constructPlayfairTable(char *key) {

    int i, j; preparePlayfairKey(key);


    printf("\nPlayfair Key Matrix:\n"); for
    (i = 0; i < SIZE; i++) {

        for (j = 0; j < SIZE; j++) {

            printf("%c ", playfair[i][j]);

        }

        printf("\n");

    }
```

```
}

void encryptPlayfair(char *text, char *key) {
    constructPlayfairTable(key);

    int i, j, a, b, m, n;
    char p1, p2;

    for (i = 0; i < strlen(text); i += 2) {
        p1 = text[i]; p2 = text[i + 1]; for (j
        = 0; j < SIZE; j++) { for (m = 0; m <
        SIZE; m++) { if (playfair[j][m] ==
        p1) {
                a = j;
                b = m;
            } else if (playfair[j][m] == p2) { a =
                j;
                b = m;
            }
        }
        if (a == 0) {
            p1 = playfair[0][b]; p2 =
            playfair[SIZE - 1][b];
        } else if (b == 0) {
            p1 = playfair[a][0]; p2 =
            playfair[a][SIZE - 1];
        } else { p1 = playfair[a][b
        - 1]; p2 = playfair[a -
        1][b]; } printf("%c%c ",
        p1, p2);
    }
    printf("\n");
}

void decryptPlayfair(char *text, char *key) {
    constructPlayfairTable(key);

    int i, j, a, b, m, n;
    char p1, p2;
```

```
    for (i = 0; i < strlen(text); i += 2) {
        p1 = text[i]; p2 = text[i + 1]; for (j
        = 0; j < SIZE; j++) { for (m = 0; m <
        SIZE; m++) { if (playfair[j][m] ==
        p1) {
                a = j;
                b = m;
            } else if (playfair[j][m] == p2) { a =
                j;
                b = m;
            }
        } if (a == SIZE - 1) { p1 =
        playfair[SIZE - 1][b];
            p2 = playfair[0][b];
        } else if (b == SIZE - 1) { p1 =
            playfair[a][SIZE - 1]; p2 =
            playfair[a][0];
        } else { p1 = playfair[a][b
            + 1]; p2 = playfair[a +
            1][b];
        }
        printf("%c%c ", p1, p2);
    }
    printf("\n");
}

int main() { char
    text[100], key[25]; int
    choice;

    printf("Enter the key (no spaces, all uppercase): "); scanf("%s",
    key);

    printf("Enter the text (uppercase): "); scanf("%s",
    text);

    printf("\n1. Encrypt\n2. Decrypt\nEnter your choice: "); scanf("%d",
    &choice);

    switch (choice) {
```

```
        case 1:
            printf("\nEncrypted Text: ");
            encryptPlayfair(text, key); break;
        case 2:
            printf("\nDecrypted Text: ");
            decryptPlayfair(text, key); break;
        default:
            printf("\nInvalid choice!\n");
    }
    return 0;
}
```

Input and Output:



Result: A c program to perform encryption and decryption of a message using Play fair substitution technique is successfully executed.