Assignment – 6

Q1. Write a program to encrypt and decrypt the message "Meet Me at the Bridge" using Play fair cipher where key is "Your Name".

A:

Pseudocode and Explanation –

1. **generating_key_matrix(string key)**

- **Purpose**: Creates a 5x5 matrix for the Playfair cipher using the given key. It avoids duplicate letters and excludes 'j'.
- **How it works**: The function first adds unique letters from the key to the matrix, then fills it with the rest of the alphabet (excluding 'j'), ensuring no letter repeats.

2. add_x(string plain_text)

- **Purpose**: Ensures the plaintext is ready for encryption by inserting 'x' between repeated letters and making the length even.
- **How it works**: It checks pairs of characters, inserts 'x' if two consecutive characters are the same, and appends 'x' if the final string has an odd length.

3. get_index(vector<vector<char>> matrix, char c)

- **Purpose**: Finds the row and column of character c in the matrix.
- **How it works**: Iterates through the matrix to locate the character and returns its position as a pair of integers.

4. main()

- **Purpose**: Performs encryption and decryption using the Playfair cipher.
- **How it works**:
    - **Key Matrix Generation**: Calls generating_key_matrix to create the matrix.

- **Plaintext Preprocessing**: Prepares the plaintext using add_x.
- **Encryption**: For each pair of characters, follows Playfair cipher rules (same row, same column, or rectangle) to generate the ciphertext.
- **Decryption**: Reverses the encryption process to obtain the original text.

Code -

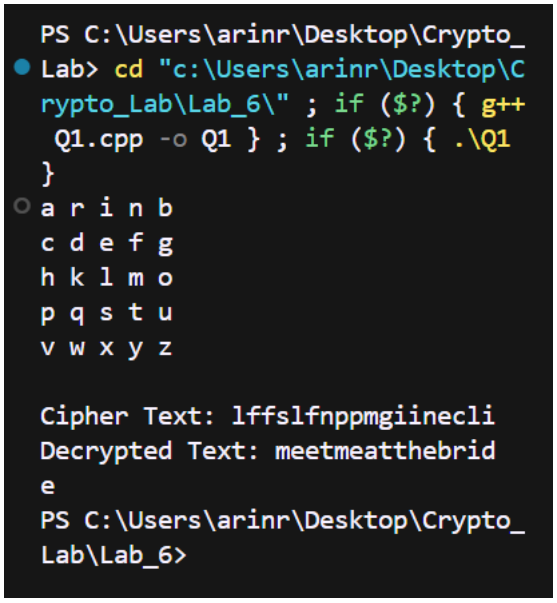
```
#include<bits/stdc++.h>
using namespace std;
vector<vector<char>> generating_key_matrix(string key){

    unordered_map<char, bool> check_occurence;
    vector<vector<char>> matrix;
    int p = 0;
    vector<char> temp;

    for(int i = 0; i < key.length(); i++){
        if(!check_occurence[key[i]]){
            check_occurence[key[i]] = true;

            temp.push_back(key[i]);
            p++;

            if(p % 5 == 0){
                matrix.push_back(temp);
                temp.clear();
            }
        }
    }


    for(int i = 0; i < 26; i++){
        if(i == 9){
            continue;
        }

        char current_char = char(i + 97);

        if(!check_occurence[current_char]){
            temp.push_back(current_char);
            p++;
```

```
            if(p % 5 == 0){
                matrix.push_back(temp);
                temp.clear();
            }
        }
    }


    if(!temp.empty()){
        matrix.push_back(temp);
    }

    return matrix;

}
string add_x(string plain_text){

    int i=0;
    string x = "x";
    while(i!=plain_text.length()){

        char p1 = plain_text[i];
        char p2;
        if((i+1) != plain_text.length()){
            p2 = plain_text[i+1];
        }else{
            plain_text.insert(i+1,x);
        }

        if(p1 == p2){
            plain_text.insert(i+1,x);
            i=0;
        }else{
            i+=2;
        }

    }

    return plain_text;
}
pair<int,int> get_index(vector<vector<char>> matrix, char c){

    for(int i=0;i<matrix.size();i++){
        for(int j=0;j<matrix[i].size();j++){
            if(c == matrix[i][j]){
                pair<int,int> p = {i,j};
                return p;
            }
```

```
        }
    }
}
int main()
{
    string key = "arin";
    vector<vector<char>> matrix = generating_key_matrix(key);

    for(int i = 0; i < matrix.size(); i++){
        for(int j = 0; j < matrix[i].size(); j++){
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
    cout<<endl;

    string plain_text = "meetmeatthebridge";
    string new_plain_text = add_x(plain_text);

    // cout<<new_plain_text<<endl;

    int i=0;
    string cipher_text;
    while(i!= new_plain_text.length()){
        char c1 = new_plain_text[i];
        char c2 = new_plain_text[i+1];
        pair<int,int> p1 = get_index(matrix, c1);
        pair<int,int> p2 = get_index(matrix, c2);
        int i1 = p1.first;
        int j1 = p1.second;
        int i2 = p2.first;
        int j2 = p2.second;
        // cout<<i1<<" "<<j1<<" "<<i2<<" "<<j2<<endl;

        char new_c1;
        char new_c2;
        if(i1 == i2){
            j1 = (j1+1)%5;
            j2 = (j2+1)%5;
            new_c1 = matrix[i1][j1];
            new_c2 = matrix[i2][j2];
        }else if(j1 == j2){
            i1 = (i1+1)%5;
            i2 = (i2+1)%5;
            new_c1 = matrix[i1][j1];
            new_c2 = matrix[i2][j2];
        }else{
            new_c1 = matrix[i1][j2];
```

```
            new_c2 = matrix[i2][j1];
        }

        cipher_text.push_back(new_c1);
        cipher_text.push_back(new_c2);
        i+=2;
    }

    cout<<"Cipher Text: "<<cipher_text<<endl;

    string decrypted_text;
    int j=0;
    while(j!= cipher_text.length()){
        char c1 = cipher_text[j];
        char c2 = cipher_text[j+1];
        pair<int,int> p1 = get_index(matrix, c1);
        pair<int,int> p2 = get_index(matrix, c2);
        int i1 = p1.first;
        int j1 = p1.second;
        int i2 = p2.first;
        int j2 = p2.second;
        // cout<<i1<<" "<<j1<<" "<<i2<<" "<<j2<<endl;

        char new_c1;
        char new_c2;
        if(i1 == i2){
            j1 = (j1-1)%5;
            j2 = (j2-1)%5;
            new_c1 = matrix[i1][j1];
            new_c2 = matrix[i2][j2];
        }else if(j1 == j2){
            i1 = (i1-1)%5;
            i2 = (i2-1)%5;
            new_c1 = matrix[i1][j1];
            new_c2 = matrix[i2][j2];
        }else{
            new_c1 = matrix[i1][j2];
            new_c2 = matrix[i2][j1];
        }

        decrypted_text.push_back(new_c1);
        decrypted_text.push_back(new_c2);
        j+=2;
    }

    cout<<"Decrypted Text: "<<decrypted_text<<endl;
}
```

Output –



Q2. Write a program to encrypt and decrypt the message "Pay more money" using trigraph Hill Cipher where key is "GYBNQKURP".

A:

Pseudocode and Explanation –

1 **Key Matrix**:

- The key string ("rrfvsvcct") is converted into a 3x3 matrix by subtracting 97 from each character's ASCII value (to represent the letters' positions in the alphabet).

2 **Plaintext Padding**:

- The plaintext ("paymoremoney") is padded with 'x' characters so that its length becomes divisible by 3.

3 **Encryption**:

- For each 3-character block in the plaintext:
    - It is treated as a vector and multiplied by the key matrix.

- The result is taken modulo 26 (to keep it within the alphabet range), and the corresponding characters are generated for the ciphertext.

4 **Decryption**:

- **Determinant and Inverse Calculation**:
  - The determinant of the key matrix is calculated using the determinant function.
  - The modular inverse of the determinant modulo 26 is found using the find_MI function.
- **Cofactor and Inverse Matrix**:
  - The cofactors of the key matrix are calculated and transposed to find the adjugate matrix.
  - This adjugate matrix is multiplied by the modular inverse of the determinant and reduced modulo 26 to produce the inverse matrix.

5 **Decryption Process**:

- The ciphertext is decrypted by multiplying the inverse key matrix with each block of the ciphertext (same process as encryption, but using the inverse key matrix).

Code –

```
#include<bits/stdc++.h>
using namespace std;
int determinant(int n , vector<vector<int>> a) {
    int det = (a[0][0]*(a[1][1]*a[2][2] - a[1][2]*a[2][1])) -
(a[0][1]*(a[1][0]*a[2][2] - a[2][0]*a[1][2])) + (a[0][2]*(a[1][0]*a[2][1] -
a[2][0]*a[1][1]));
    return det;
}
int find_MI(int x1, int x2){

    bool x = false;
```

```
        int c = 1;
        while(x != true){
            if((x1*c)%x2 == 1){
                x = true;
            }else{
                c++;
            }
        }
    return c;
}
int main()
{
    string key = "gybnqkurp";
    // string key = "rrfvsvcct";

    vector<vector<int>> key_matrix;
    int p=0;
    vector<int> temp;
    for(int i=0;i<key.length();i++){
        temp.push_back(int(key[i])-97);
        p++;
        if(p%3 == 0){
            key_matrix.push_back(temp);
            temp.clear();
        }
    }

    // for(int i=0;i<key_matrix.size();i++){
    //     for(int j=0;j<key_matrix[i].size();j++){
    //         cout<<key_matrix[i][j]<<" ";
    //     }
    //     cout<<endl;
    // }

    string plain_text = "paymoremoney";
    if((plain_text.length())%3 == 1){
        plain_text.push_back('x');
        plain_text.push_back('x');
    }else if((plain_text.length())%3 == 2){
        plain_text.push_back('x');
    }
    // cout<<plain_text;

    int i=0;
    string cipher_text;
    while(i!=plain_text.length()){
        int c1 = int(plain_text[i]-97);
        int c2 = int(plain_text[i+1]-97);
```

```
        int c3 = int(plain_text[i+2]-97);

        for(int j=0;j<3;j++){
            int val = c1*key_matrix[0][j] + c2*key_matrix[1][j] +
c3*key_matrix[2][j];
            int val2 = (val%26) +97;
            cipher_text.push_back(char(val2));
        }
        i+=3;
    }

    cout<<"Cipher Text: "<<cipher_text<<endl;

    // Decryption
    int n = key_matrix.size();
    int det = determinant(n, key_matrix);
    if(det < 0){
        int x = det*-1;
        det = 26 - (x%26);
    }

    int det_inv = find_MI(det,26);
    vector<vector<int>> cof;

    for(int i=0;i<n;i++){
        vector<int> temp;
        for(int j=0;j<n;j++){
            int x = i;
            int y = j;
            int x1,x2;
            int y1,y2;
            if(x == 0){
                x1 = 1;
                x2 = 2;
            }else if(x == 1){
                x1 = 0;
                x2 = 2;
            }else{
                x1 = 0;
                x2 = 1;
            }

            if(y == 0){
                y1 = 1;
                y2 = 2;
            }else if(y == 1){
                y1 = 0;
                y2 = 2;
```

```
            }else{
                y1 = 0;
                y2 = 1;
            }
            int val = key_matrix[x1][y1]*key_matrix[x2][y2] -
key_matrix[x1][y2]*key_matrix[x2][y1];
            if((x+y)%2 != 0){
                val = val*-1;
            }
            temp.push_back(val);
        }
        cof.push_back(temp);
    }


    // Transpose matrix
    vector<vector<int>> k_inverse;

    for(int i=0;i<n;i++){
        vector<int> temp;
        for(int j=0;j<n;j++){
            temp.push_back(cof[j][i]);
        }
        k_inverse.push_back(temp);
    }

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            int fg = k_inverse[i][j];
            if(fg < 0){
                int x = fg*-1;
                fg = 26 - (x%26);
                k_inverse[i][j] = fg;
            }else{
                k_inverse[i][j] = (k_inverse[i][j])%26;
            }
        }
    }
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            int fg = k_inverse[i][j]*det_inv;
            k_inverse[i][j] = fg%26;
        }
        cout<<endl;
    }
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            cout<<k_inverse[i][j]<<" ";
```

```
        }
        cout<<endl;
    }


    string decrypted_text="";
    i=0;
    while(i!=cipher_text.length()){
        int c1 = int(cipher_text[i]-97);
        int c2 = int(cipher_text[i+1]-97);
        int c3 = int(cipher_text[i+2]-97);

        for(int j=0;j<3;j++){
            int val = c1*k_inverse[0][j] + c2*k_inverse[1][j] +
c3*k_inverse[2][j];
            int val2 = (val%26) +97;
            // cout<<val2<<" ";
            decrypted_text.push_back(char(val2));
        }
        i+=3;
    }

    cout<<"Decrypted Text: "<<decrypted_text;

}
```

Output –

```
b\Lab_6> cd "c:\Users\arinr\Desktop
\Crypto_Lab\Lab_6\" ; if ($?) { g++
 Q2.cpp -o Q2 } ; if ($?) { .\Q2 }
Cipher Text: yolwvrsgwmex



8 5 10
21 8 21
21 12 8
Decrypted Text: paymoremoney
PS C:\Users\arinr\Desktop\Crypto_La
b\Lab_6>
```