Assignment – 7

Q1. Write the Encryption and decryption procedure for

i) Column Transposition

ii) Route cipher

iii) Row column transposition

iv) Rail fence cipher

A:

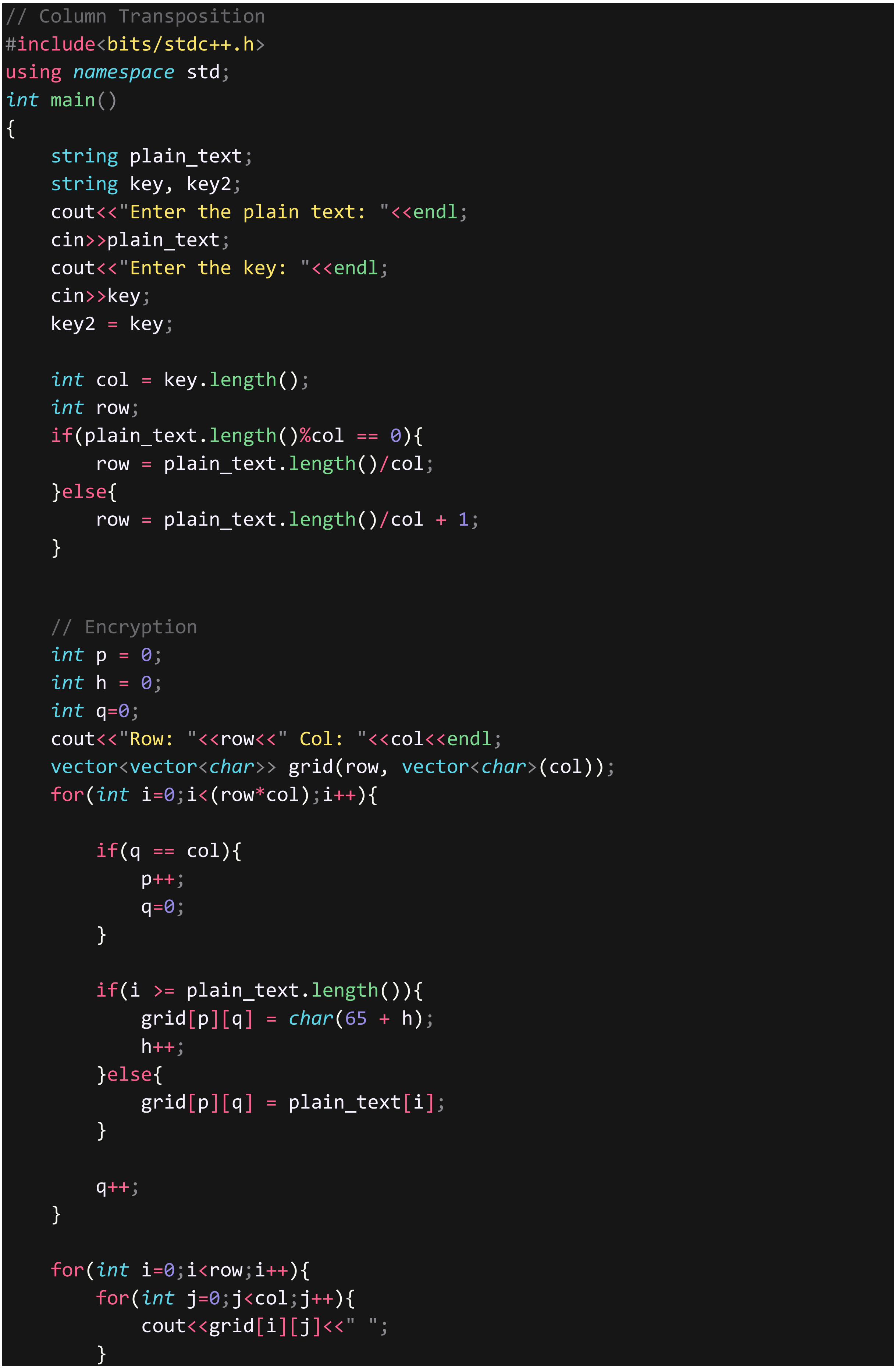
i.)

Pseudocode and Explanation –

1. **Input:** It takes a plain text and a key (used for column transposition).
2. **Grid Setup:** The plain text is arranged in a grid with rows determined by the key length. If the text doesn't fill the grid, extra characters (A, B, C, ...) are added.
3. **Encryption:**
   - Characters from the grid columns are rearranged based on the alphabetical order of the key.
   - A map stores the characters column-wise, indexed by the key.
   - The encrypted text is formed by reading characters column-wise based on the sorted key.
4. **Decryption:**
   - The process is reversed by mapping the characters back to their original columns using the original (unsorted) key.
   - The decrypted text is formed by reading characters row-wise from the grid.

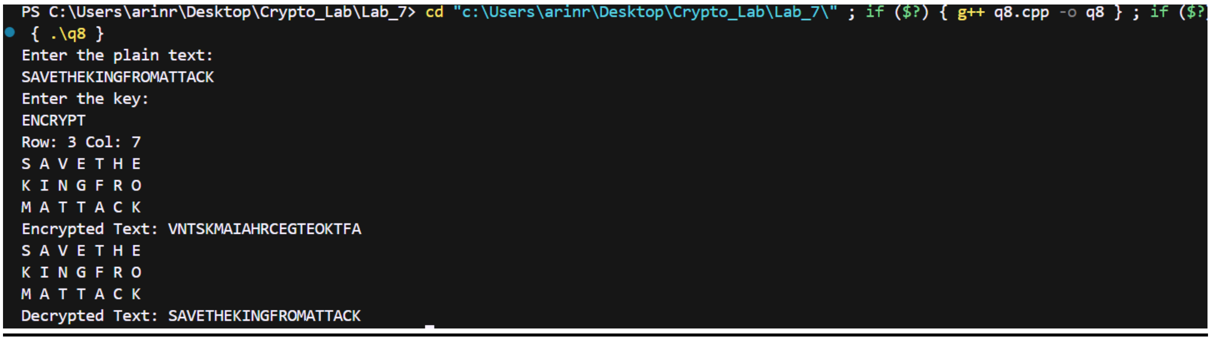
   The code outputs the encrypted and decrypted text.

Code-



```cpp
// Column Transposition
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string plain_text;
    string key, key2;
    cout<<"Enter the plain text: "<<endl;
    cin>>plain_text;
    cout<<"Enter the key: "<<endl;
    cin>>key;
    key2 = key;

    int col = key.length();
    int row;
    if(plain_text.length()%col == 0){
        row = plain_text.length()/col;
    }else{
        row = plain_text.length()/col + 1;
    }


    // Encryption
    int p = 0;
    int h = 0;
    int q=0;
    cout<<"Row: "<<row<<" Col: "<<col<<endl;
    vector<vector<char>> grid(row, vector<char>(col));
    for(int i=0;i<(row*col);i++){

        if(q == col){
            p++;
            q=0;
        }

        if(i >= plain_text.length()){
            grid[p][q] = char(65 + h);
            h++;
        }else{
            grid[p][q] = plain_text[i];
        }

        q++;
    }

    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            cout<<grid[i][j]<<" ";
        }
        cout<<endl;
    }

    unordered_map<char, vector<char>> alpha_map;
    for(int i=0;i<key.length();i++){
        for(int j=0;j<row;j++){
            alpha_map[key[i]].push_back(grid[j][i]);
        }
    }


    sort(key.begin(),key.end());

    string encrypted_text;
    for(int i=0;i<key.length();i++){
        char ch = key[i];
        for(int j=0;j<row;j++){
            encrypted_text+=alpha_map[ch][j];
        }
    }

    cout<<"Encrypted Text: "<<encrypted_text;


    // Decryption
    vector<vector<char>> grid_2(row, vector<char>(col));
    for(int i=0;i<col;i++){
        for(int j=0;j<row;j++){
            grid_2[j][i] = alpha_map[key2[i]][j];
        }
    }

    cout<<endl;
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            cout<<grid_2[i][j]<<" ";
        }
        cout<<endl;
    }

    string decrypted_text;
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            decrypted_text+=grid_2[i][j];
        }
    }
    cout<<"Decrypted Text: "<<decrypted_text;

}
```

Output –



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_7> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_7\" ; if ($?) { g++ q8.cpp -o q8 } ; if ($?
 { .\q8 }
Enter the plain text:
SAVETHEKINGFROMATTACK
Enter the key:
ENCRYPT
Row: 3 Col: 7
S A V E T H E
K I N G F R O
M A T T A C K
Encrypted Text: VNTSKMAIAHRCEGTEOKTFA
S A V E T H E
K I N G F R O
M A T T A C K
Decrypted Text: SAVETHEKINGFROMATTACK
```

ii)

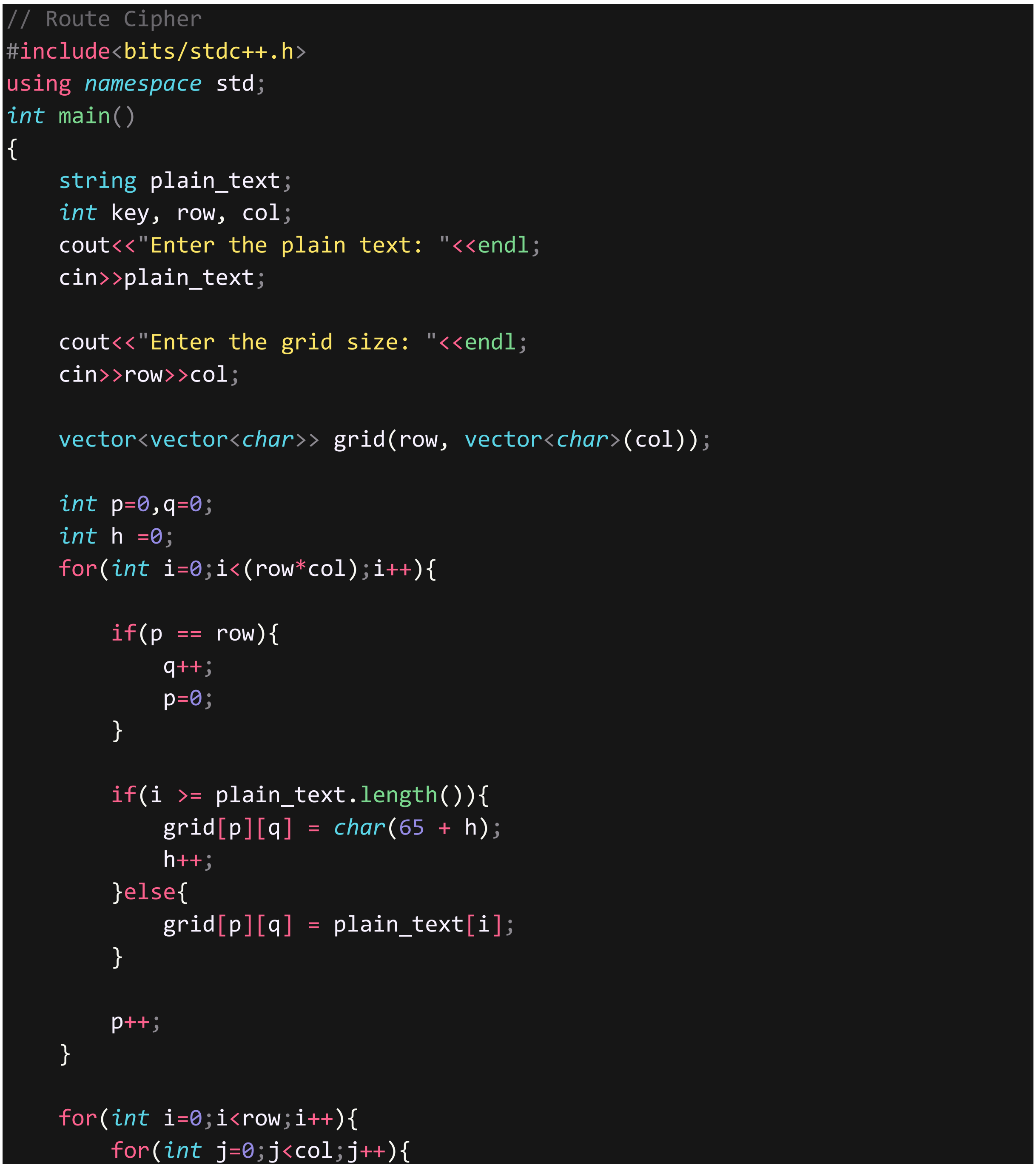
Pseudocode and Explanation –

1. **Input:**
    - The user inputs a plain text message and the grid size (row and col).
2. **Grid Setup:**
    - The plain text is arranged into a grid of dimensions row x col.
    - If the plain text doesn't fill the grid, additional characters (A, B, C, ...) are added to fill the grid.
3. **Encryption (Spiral/Route traversal):**
    - The grid is traversed in a spiral order:
        - Start from the top-right corner.
        - Move downward along the rightmost column.
        - Then, move left along the bottom row.
        - Move upward along the leftmost column.
        - Continue this pattern, spiraling inward until the entire grid is traversed.

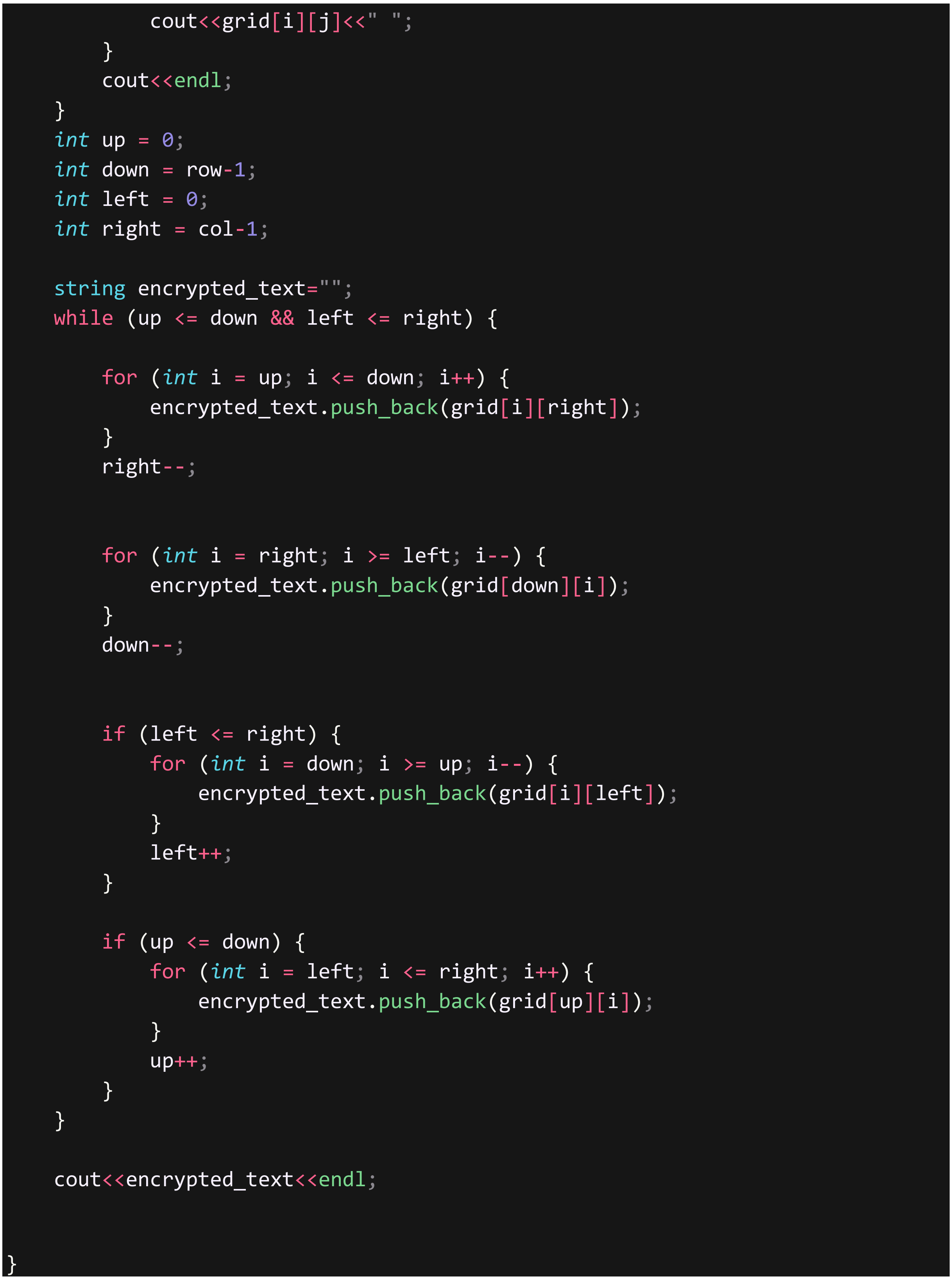
   - The characters are collected in the order they are traversed to form the encrypted text.

4. **Output:**
   - The final encrypted text is printed after completing the spiral traversal of the grid.

Code-

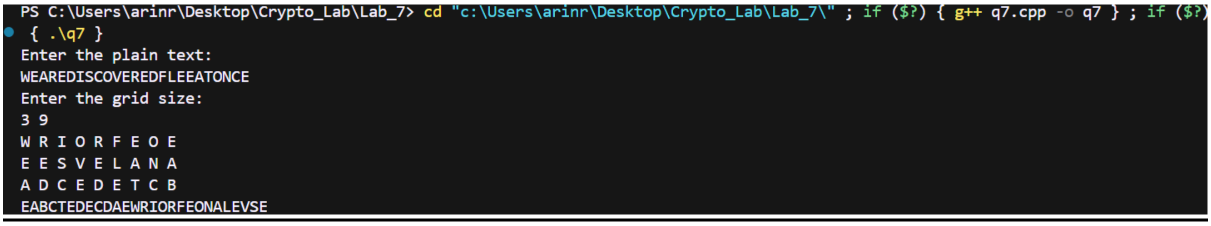


```
// Route Cipher
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string plain_text;
    int key, row, col;
    cout<<"Enter the plain text: "<<endl;
    cin>>plain_text;

    cout<<"Enter the grid size: "<<endl;
    cin>>row>>col;

    vector<vector<char>> grid(row, vector<char>(col));

    int p=0,q=0;
    int h =0;
    for(int i=0;i<(row*col);i++){

        if(p == row){
            q++;
            p=0;
        }

        if(i >= plain_text.length()){
            grid[p][q] = char(65 + h);
            h++;
        }else{
            grid[p][q] = plain_text[i];
        }

        p++;
    }

    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
```



```
            cout<<grid[i][j]<<" ";
        }
        cout<<endl;
    }
    int up = 0;
    int down = row-1;
    int left = 0;
    int right = col-1;

    string encrypted_text="";
    while (up <= down && left <= right) {

        for (int i = up; i <= down; i++) {
            encrypted_text.push_back(grid[i][right]);
        }
        right--;


        for (int i = right; i >= left; i--) {
            encrypted_text.push_back(grid[down][i]);
        }
        down--;


        if (left <= right) {
            for (int i = down; i >= up; i--) {
                encrypted_text.push_back(grid[i][left]);
            }
            left++;
        }

        if (up <= down) {
            for (int i = left; i <= right; i++) {
                encrypted_text.push_back(grid[up][i]);
            }
            up++;
        }
    }

    cout<<encrypted_text<<endl;


}
```

Output –



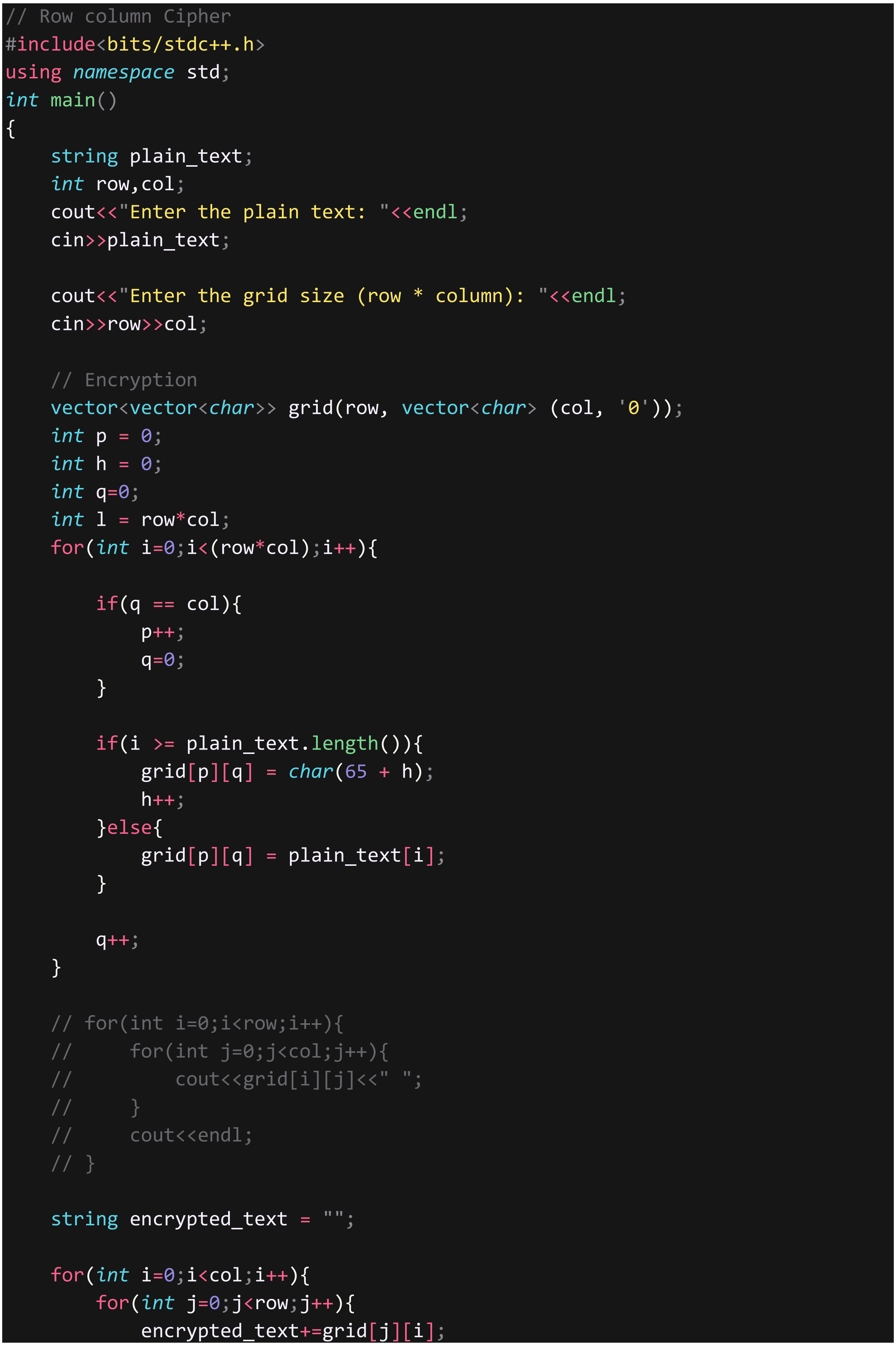
```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_7> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_7\" ; if ($?) { g++ q7.cpp -o q7 } ; if ($?)
 { .\q7 }
Enter the plain text:
WEAREDISCOVEREDFLEEATONCE
Enter the grid size:
3 9
W R I O R F E O E
E E S V E L A N A
A D C E D E T C B
EABCTEDECDAEWRIORFEONALEVSE
```

iii)

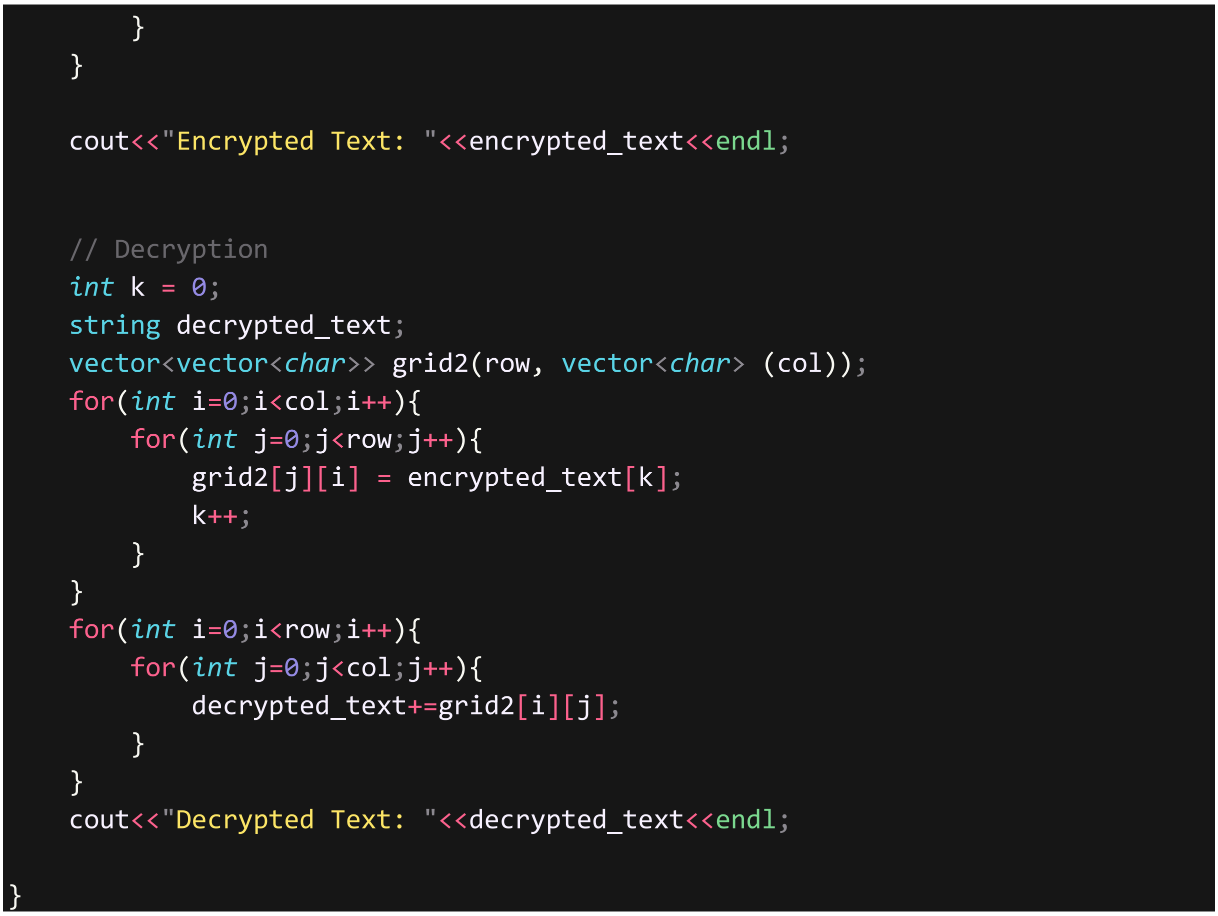
Pseudocode and Explanation –

1. **Input:**
   - The user inputs a plain text and specifies the grid size (number of rows and columns).
2. **Encryption:**
   - The plain text is placed into a grid row by row.
   - If the grid size exceeds the text length, filler characters (A, B, C, ...) are added to fill the grid.
   - The encrypted text is generated by reading the grid column by column (instead of row by row).
   - The encrypted text is output.
3. **Decryption:**
   - The encrypted text is placed back into a new grid column by column.
   - The decrypted text is then generated by reading the grid row by row, reversing the column-wise arrangement of the encryption process.
4. **Output:**
   - The program outputs the encrypted text and then the decrypted text, which should match the original plain text.

Code-

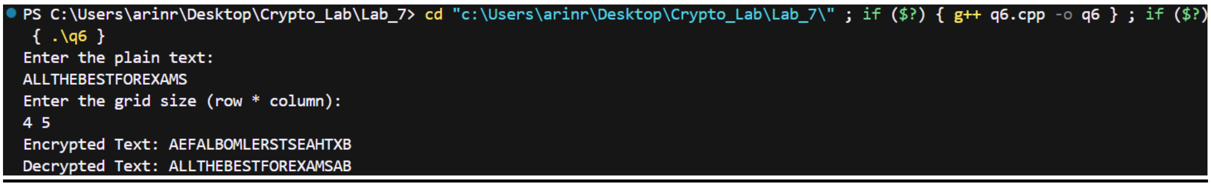


```cpp
// Row column Cipher
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string plain_text;
    int row,col;
    cout<<"Enter the plain text: "<<endl;
    cin>>plain_text;

    cout<<"Enter the grid size (row * column): "<<endl;
    cin>>row>>col;

    // Encryption
    vector<vector<char>> grid(row, vector<char> (col, '0'));
    int p = 0;
    int h = 0;
    int q=0;
    int l = row*col;
    for(int i=0;i<(row*col);i++){

        if(q == col){
            p++;
            q=0;
        }

        if(i >= plain_text.length()){
            grid[p][q] = char(65 + h);
            h++;
        }else{
            grid[p][q] = plain_text[i];
        }

        q++;
    }

    // for(int i=0;i<row;i++){
    //     for(int j=0;j<col;j++){
    //         cout<<grid[i][j]<<" ";
    //     }
    //     cout<<endl;
    // }

    string encrypted_text = "";

    for(int i=0;i<col;i++){
        for(int j=0;j<row;j++){
            encrypted_text+=grid[j][i];
```



```
        }
    }

    cout<<"Encrypted Text: "<<encrypted_text<<endl;


    // Decryption
    int k = 0;
    string decrypted_text;
    vector<vector<char>> grid2(row, vector<char> (col));
    for(int i=0;i<col;i++){
        for(int j=0;j<row;j++){
            grid2[j][i] = encrypted_text[k];
            k++;
        }
    }
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            decrypted_text+=grid2[i][j];
        }
    }
    cout<<"Decrypted Text: "<<decrypted_text<<endl;

}
```

Output –



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_7> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_7\" ; if ($?) { g++ q6.cpp -o q6 } ; if ($?)
 { .\q6 }
Enter the plain text:
ALLTHEBESTFOREXAMS
Enter the grid size (row * column):
4 5
Encrypted Text: AEFALBOMLERSTSEAHTXB
Decrypted Text: ALLTHEBESTFOREXAMSAB
```

iv)

Pseudocode and Explanation –

1. **Input:**

- The user inputs a plain text and a key (the number of rails/rows).

2. **Grid Setup (Zigzag Pattern):**

- A grid with key rows and a number of columns equal to the length of the plain text is created.
- The plain text is written in a zigzag pattern across the rows:
    - Characters are placed on the rails going down and then up repeatedly, mimicking a wave or zigzag.
- Each letter from the plain text is placed into the appropriate row of the grid based on the zigzag pattern.

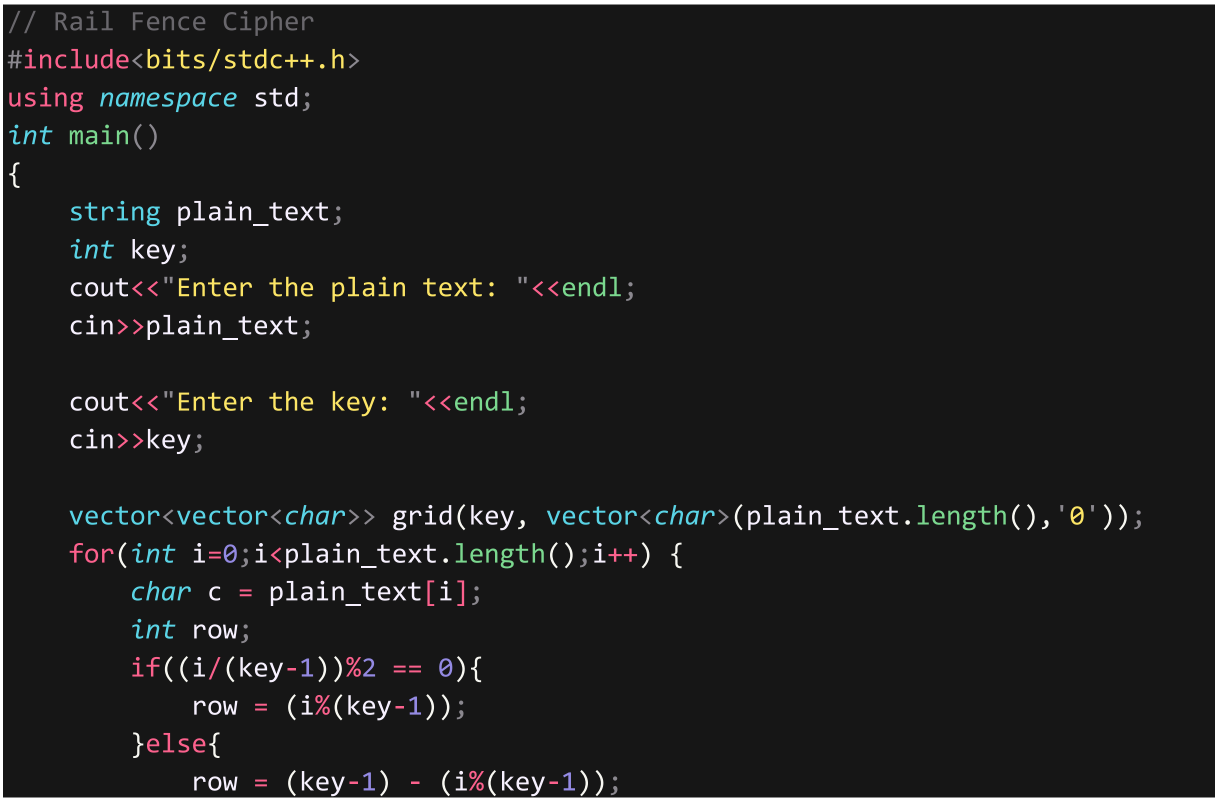
3. **Encryption:**

- After filling the grid in the zigzag manner, the encrypted text is generated by reading the characters row by row (rails), ignoring empty positions in the grid.

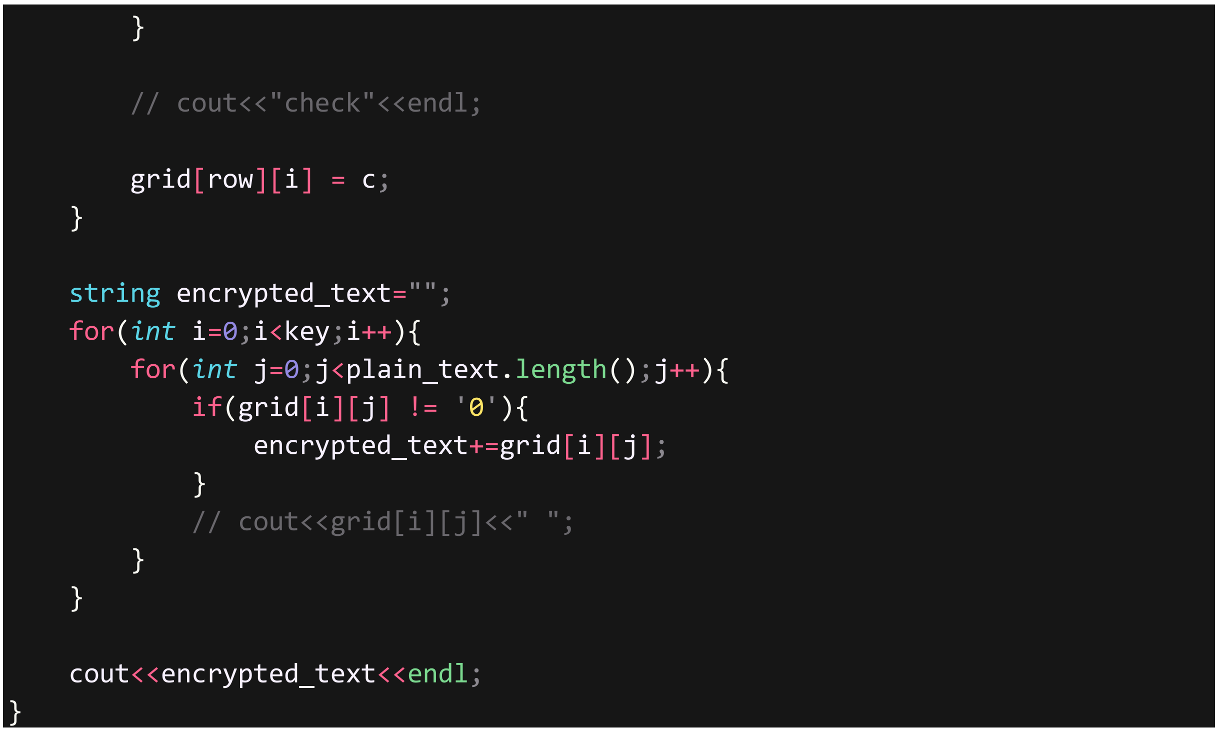
4. **Output:**

- The encrypted text is printed as the result of reading all the rows in sequence.

Code-

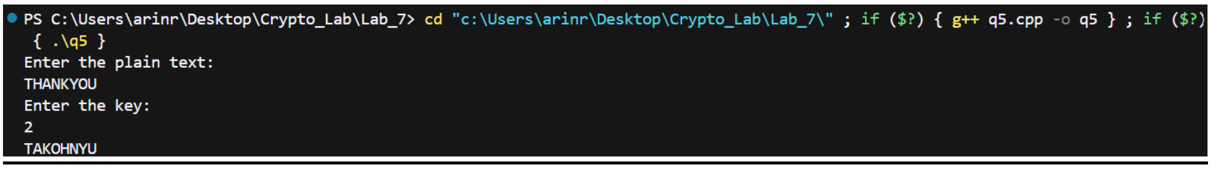


```
// Rail Fence Cipher
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string plain_text;
    int key;
    cout<<"Enter the plain text: "<<endl;
    cin>>plain_text;

    cout<<"Enter the key: "<<endl;
    cin>>key;

    vector<vector<char>> grid(key, vector<char>(plain_text.length(),'0'));
    for(int i=0;i<plain_text.length();i++) {
        char c = plain_text[i];
        int row;
        if((i/(key-1))%2 == 0){
            row = (i%(key-1));
        }else{
            row = (key-1) - (i%(key-1));
```



```
        }

        // cout<<"check"<<endl;

        grid[row][i] = c;
    }

    string encrypted_text="";
    for(int i=0;i<key;i++){
        for(int j=0;j<plain_text.length();j++){
            if(grid[i][j] != '0'){
                encrypted_text+=grid[i][j];
            }
            // cout<<grid[i][j]<<" ";
        }
    }

    cout<<encrypted_text<<endl;
}
```

Output-



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_7> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_7\" ; if ($?) { g++ q5.cpp -o q5 } ; if ($?)
 { .\q5 }
Enter the plain text:
THANKYOU
Enter the key:
2
TAKOHNYU
```

Q2. Write a program to encrypt and decrypt the text using Affine cipher.

A:

Pseudocode and Explanation –

1. **Input:**

- The user inputs a plain text (which should be in uppercase letters) and two parameters, a and b, which are used in the encryption formula.

2. **Finding Multiplicative Inverse:**

- The function find_MI(int a, int n) calculates the multiplicative inverse of a modulo n (in this case, n is 26, corresponding to the letters of the English alphabet). This is crucial for decryption.

- It uses the Extended Euclidean Algorithm to find the inverse.

3. **Encryption:**

- Each character of the plain text is converted into a numerical value (A=0, B=1, ..., Z=25).
- The encryption formula E(x)=(a·p+b)mod 26E(x) = (a \cdot p + b) \mod 26E(x)=(a·p+b)mod26 is applied to each character, where ppp is the numerical value of the character.
- The resulting values are converted back to characters to form the encrypted text.

4. **Output Encrypted Text:**

- The encrypted text is printed.

5. **Decryption:**

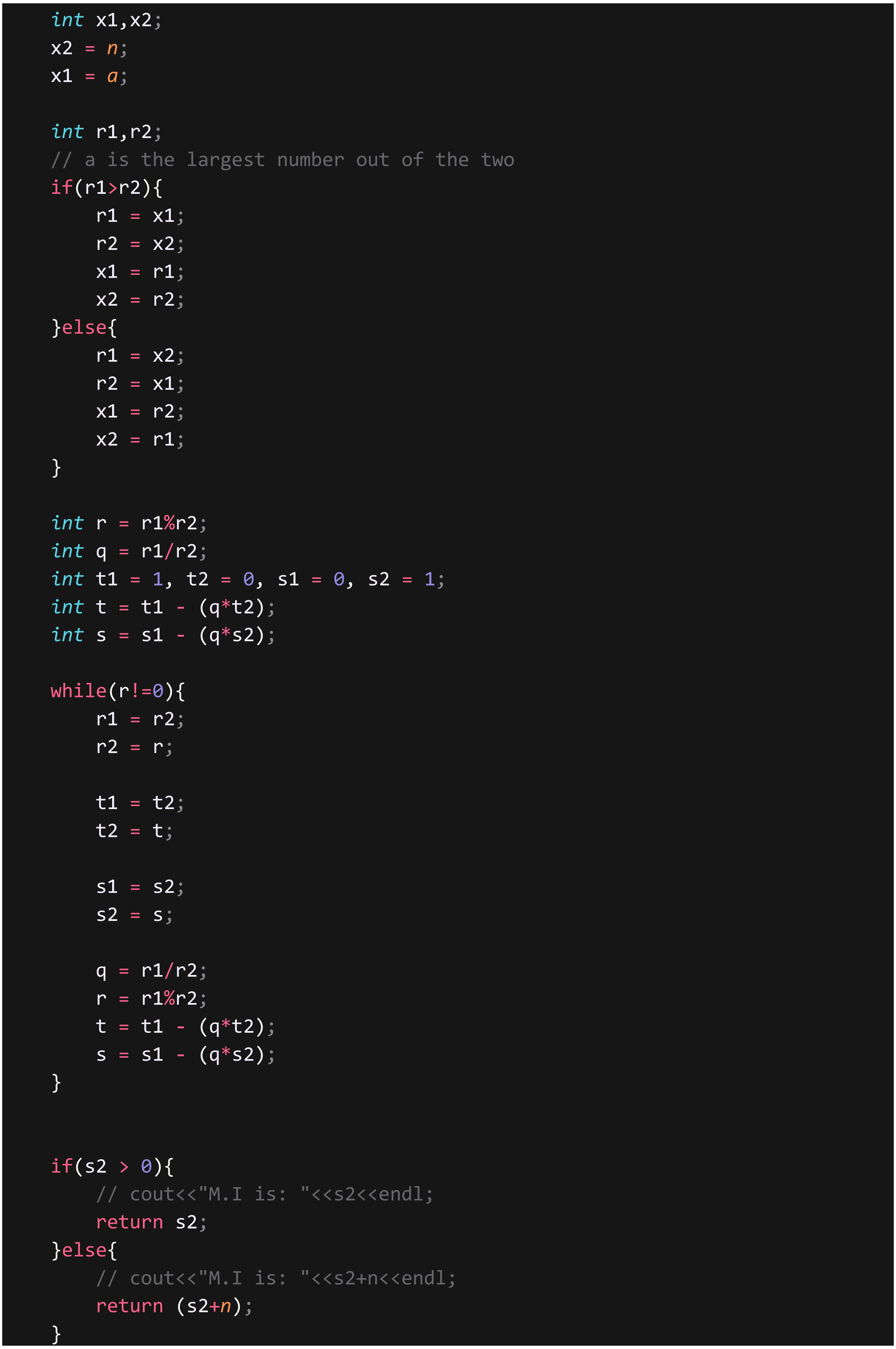
- To decrypt, the multiplicative inverse of a is found using the find_MI function.
- The decryption formula D(c)=(mi·(c−b))mod 26D(c) = (mi \cdot (c - b)) \mod 26D(c)=(mi·(c−b))mod26 is applied to each character of the encrypted text, where ccc is the numerical value of the encrypted character.
- The resulting values are converted back to characters to form the decrypted text.

6. **Output Decrypted Text:**

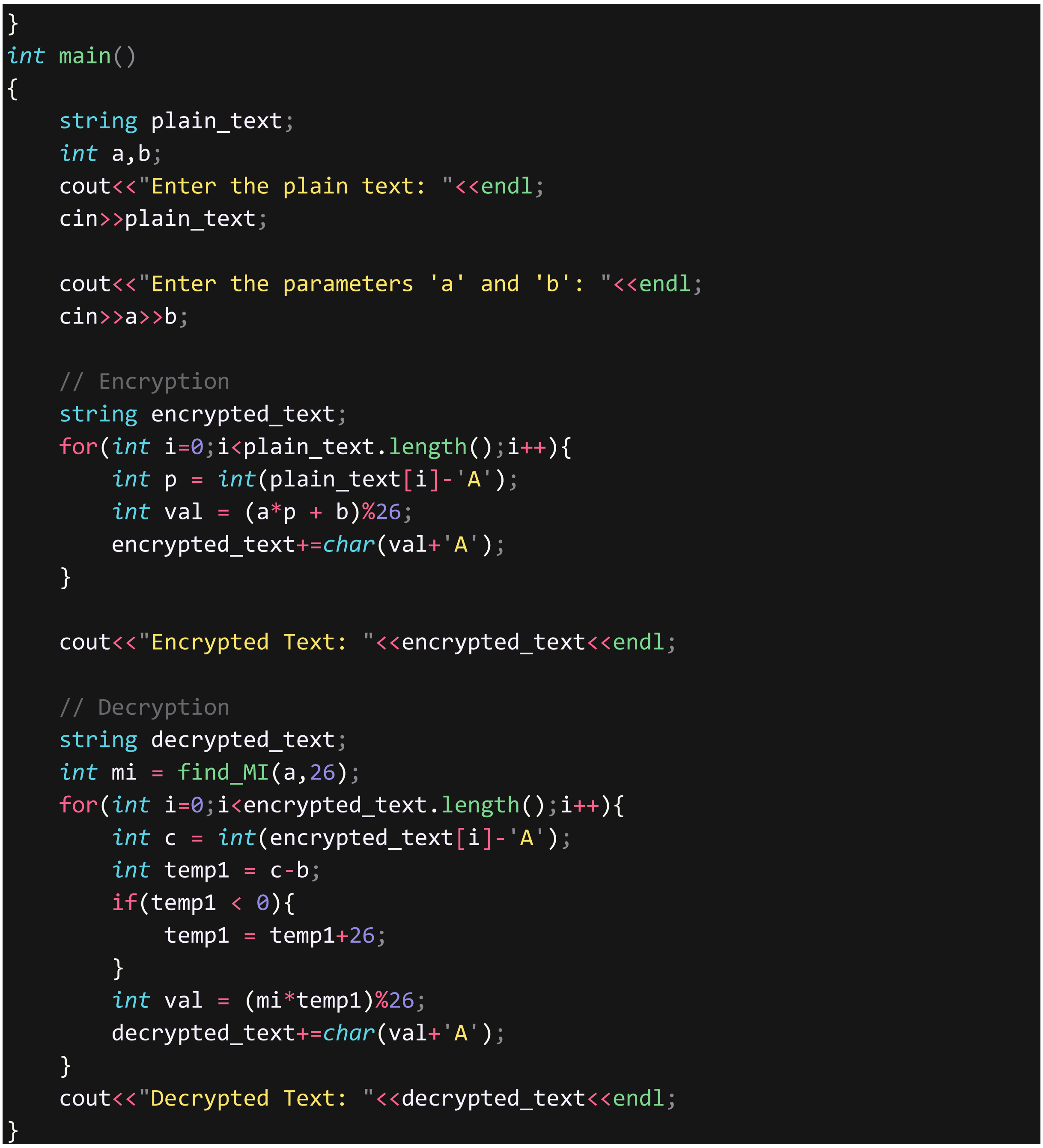
- The decrypted text is printed, which should match the original plain text if everything is done correctly.

Code-

```
// Affine Cipher
#include<bits/stdc++.h>
using namespace std;

int find_MI(int a,int n){
```

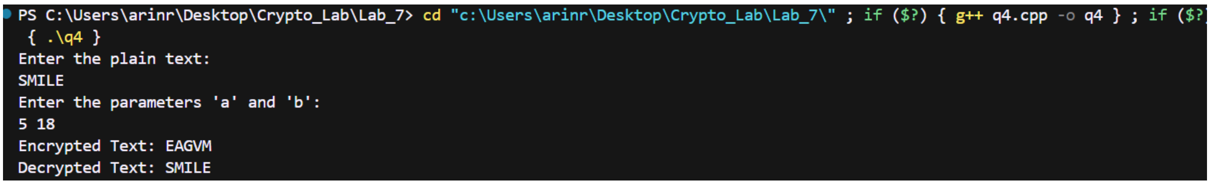


```
int x1,x2;
x2 = n;
x1 = a;

int r1,r2;
// a is the largest number out of the two
if(r1>r2){
    r1 = x1;
    r2 = x2;
    x1 = r1;
    x2 = r2;
}else{
    r1 = x2;
    r2 = x1;
    x1 = r2;
    x2 = r1;
}

int r = r1%r2;
int q = r1/r2;
int t1 = 1, t2 = 0, s1 = 0, s2 = 1;
int t = t1 - (q*t2);
int s = s1 - (q*s2);

while(r!=0){
    r1 = r2;
    r2 = r;

    t1 = t2;
    t2 = t;

    s1 = s2;
    s2 = s;

    q = r1/r2;
    r = r1%r2;
    t = t1 - (q*t2);
    s = s1 - (q*s2);
}


if(s2 > 0){
    // cout<<"M.I is: "<<s2<<endl;
    return s2;
}else{
    // cout<<"M.I is: "<<s2+n<<endl;
    return (s2+n);
}
```



```
}
int main()
{
    string plain_text;
    int a,b;
    cout<<"Enter the plain text: "<<endl;
    cin>>plain_text;

    cout<<"Enter the parameters 'a' and 'b': "<<endl;
    cin>>a>>b;

    // Encryption
    string encrypted_text;
    for(int i=0;i<plain_text.length();i++){
        int p = int(plain_text[i]-'A');
        int val = (a*p + b)%26;
        encrypted_text+=char(val+'A');
    }

    cout<<"Encrypted Text: "<<encrypted_text<<endl;

    // Decryption
    string decrypted_text;
    int mi = find_MI(a,26);
    for(int i=0;i<encrypted_text.length();i++){
        int c = int(encrypted_text[i]-'A');
        int temp1 = c-b;
        if(temp1 < 0){
            temp1 = temp1+26;
        }
        int val = (mi*temp1)%26;
        decrypted_text+=char(val+'A');
    }
    cout<<"Decrypted Text: "<<decrypted_text<<endl;
}
```

Output-



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_7> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_7\" ; if ($?) { g++ q4.cpp -o q4 } ; if ($?
 { .\q4 }
Enter the plain text:
SMILE
Enter the parameters 'a' and 'b':
5 18
Encrypted Text: EAGVM
Decrypted Text: SMILE
```

Q3. Write a program to encrypt and decrypt the text using vigenere and vernam cipher.

A:

Pseudocode and Explanation – Vigenere Cipher

1. **Input:**

- The user inputs a plain text (which should be in uppercase letters) and a key (also in uppercase letters).

2. **Encryption:**

- The length of the key is determined.
- For each character in the plain text:
    - The corresponding character from the key is determined using modulo arithmetic to repeat the key as necessary.
    - The encrypted character is calculated using the formula: cipher_char=(plain_char+key_char)mod26
    - The result is converted back to a character and stored in a cipher string.

3. **Output Encrypted Text:**

- The encrypted text is printed.

4. **Decryption:**

- For each character in the cipher text:
    - The corresponding character from the key is used again.
    - The decrypted character is calculated using the formula: plain_char=(cipher_char−key_char+26)mod26
    - The result is converted back to a character and stored in a plain string.

5. **Output Decrypted Text:**

- The decrypted text is printed, which should match the original plain text if everything is executed correctly.

<u>Pseudocode and Explanation – Vernom Cipher</u>

1. **Input:**

- The user inputs a plain text (in uppercase letters) and a key (also in uppercase letters).

2. **Key Length Check:**

- The code checks if the length of the plain text is equal to the length of the key. If they do not match, it prompts the user to enter a valid key.

3. **Encryption:**

- For each character in the plain text:
    - The corresponding character from the key is taken.
    - Both characters are converted to numerical values (0-25 corresponding to A-Z).
    - The XOR operation is applied between the two values: $x=x1\oplus x2$
    - The result is then reduced modulo 26 to ensure it wraps around within the alphabet.
    - The resulting value is converted back to a character and added to the encrypted text.

4. **Output Encrypted Text:**

- The encrypted text is printed.

5. **Decryption:**

- The decryption process is similar to the encryption process:
    - The same XOR operation is applied using the encrypted text and the key to retrieve the original plain text.

- The result is converted back to characters and added to the decrypted text.

6. **Output Decrypted Text:**

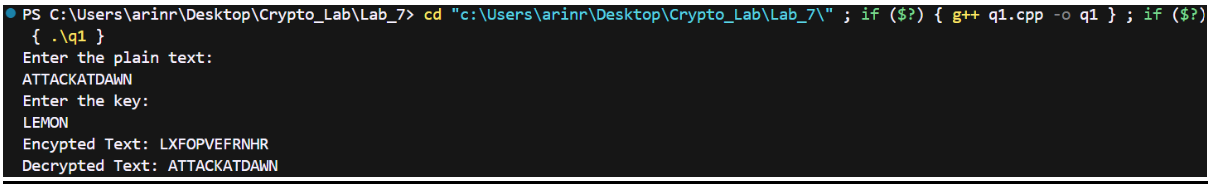
- The decrypted text is printed, which should match the original plain text if everything is executed correctly.

Code-

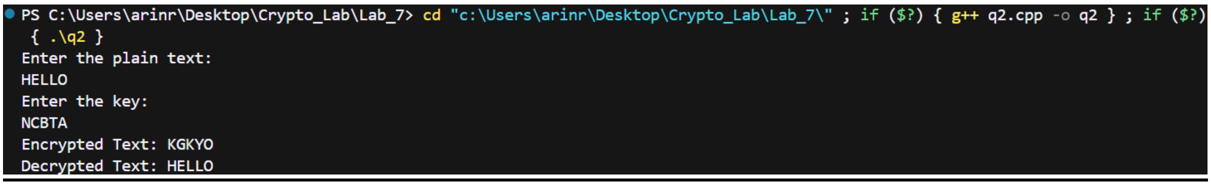
```
// Vigenere Cipher
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string plain_text,key;
    cout<<"Enter the plain text: "<<endl;
    cin>>plain_text;

    cout<<"Enter the key: "<<endl;
    cin>>key;

    int len_key = key.length();

    // Encyption
    vector<int> c_text;
    string cipher;
    for(int i=0;i<plain_text.length();i++){
        int val = int(plain_text[i] - 'A') + int(key[i%len_key] - 'A');
        int val2 = val%26;
        c_text.push_back(val2);
    }

    for(int i=0;i<c_text.size();i++){
        cipher+=char(c_text[i]+'A');
    }

    cout<<"Encypted Text: "<<cipher<<endl;

    // Decryption
    string plain;
    for(int i=0;i<c_text.size();i++){
        int val = c_text[i] - int(key[i%len_key] - 'A');
        if(val<0){
            val+=26;
        }
        plain+=char(val+'A');
    }

    cout<<"Decrypted Text: "<<plain;
}
```

```
// Vernom Cipher
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string plain_text,key;
    cout<<"Enter the plain text: "<<endl;
    cin>>plain_text;

    cout<<"Enter the key: "<<endl;
    cin>>key;

    if(plain_text.length() != key.length()){
        cout<<"Enter a valid key";
    }else{

        // Encryption
        string encrypted_text;
        for(int i=0;i<plain_text.length();i++){
            int x1 = int(plain_text[i]-'A');
            int x2 = int(key[i]-'A');
            unsigned int x = x1^x2;
            x = x%26;
            encrypted_text+=char(x+'A');
        }
        cout<<"Encrypted Text: "<<encrypted_text<<endl;

        // Decryption
        string decrypted_text;
        for(int i=0;i<encrypted_text.length();i++){
            int x1 = int(encrypted_text[i]-'A');
            int x2 = int(key[i]-'A');
            unsigned int x = x1^x2;
            x = x%26;
            decrypted_text+=char(x+'A');
        }
        cout<<"Decrypted Text: "<<decrypted_text<<endl;
    }
}
```

Output-



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_7> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_7\" ; if ($?) { g++ q1.cpp -o q1 } ; if ($?)
 { .\q1 }
Enter the plain text:
ATTACKATDAWN
Enter the key:
LEMON
Encypted Text: LXFOPVEFRNHR
Decrypted Text: ATTACKATDAWN
```



```
PS C:\Users\arinr\Desktop\Crypto_Lab\Lab_7> cd "c:\Users\arinr\Desktop\Crypto_Lab\Lab_7\" ; if ($?) { g++ q2.cpp -o q2 } ; if ($?)
 { .\q2 }
Enter the plain text:
HELLO
Enter the key:
NCBTA
Encrypted Text: KGKYO
Decrypted Text: HELLO
```