Institute of Advanced Research

The University for Innovation

Name: Desai Vraj Pinakin

Enrollment No: IAR/15593

IAR University

Subject: Cryptography And Information Security (315P)

Semester: 4rth

Section: C

Program: B.Tech.CE

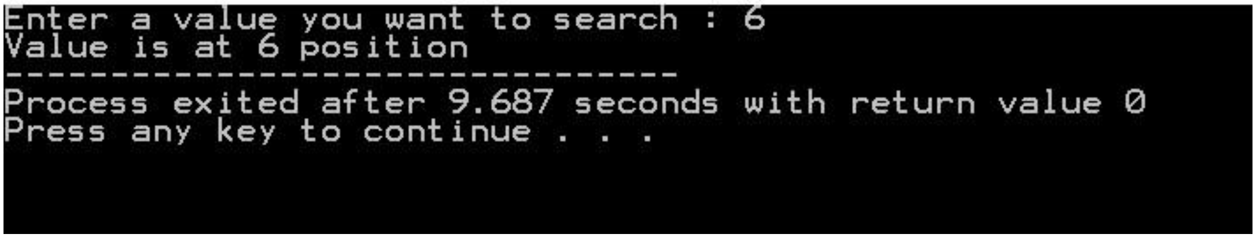
INDEX

S.No	Title	Page	Date	Sign

Practical - 1 : Linear Search

```
#include<iostream>
using namespace std; int
main()
{
        int a[10] = {1,2,3,4,5,6,7,8,9,10},val,flag=0;
cout<<"Enter a value you want to search : ";
        cin>>val;
         for(int i = 0; i < 10; i++)
        {
                if(a[i] == val)
                {
                        cout<<"Value is at "<< a[i] << " position";
                        flag = 1;
                }
        }
        if(flag == 0)
        {
                cout<<"Value is not found.";
        }
        return 0;
}
```

Output:

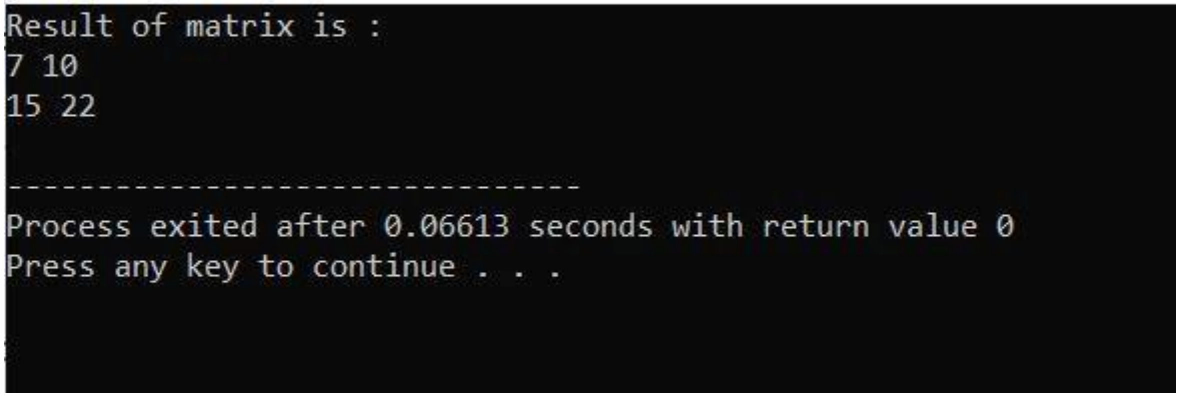


```
Enter a value you want to search : 6
Value is at 6 position
--------------------------------
Process exited after 9.687 seconds with return value 0
Press any key to continue . . .
```

Practical - 2 : Matrix Multiplication

```
#include<iostream>
using namespace std; int
main()
{
        int matrix1[2][2] = {{1,2},{3,4}},matrix2[2][2] =
{{1,2},{3,4}};      int resmatrix[2][2] = {{0,0},{0,0}};    for(int i = 0;i
< 2; i++)
        {
                for(int j = 0; j < 2; j++)
                {
                        for(int k = 0; k < 2; k++)
                        {
                                resmatrix[i][j] += matrix1[i][k]*matrix2[k][j];
                        }
                }
        }
        cout<<"Result of matrix is : "<<endl;
for(int i = 0;i < 2; i++)
        {
                for(int j = 0; j < 2; j++)
                {
                        cout<<resmatrix[i][j]<<" ";
                }
                cout<<endl;
        }
        return 0;
}
```

Output:



```
Result of matrix is :
7 10
15 22

--------------------------------
Process exited after 0.06613 seconds with return value 0
Press any key to continue . . .
```

Practical- 3: Write the logic for following conversions:

1. From decimal to octal

2. Binary to Octal

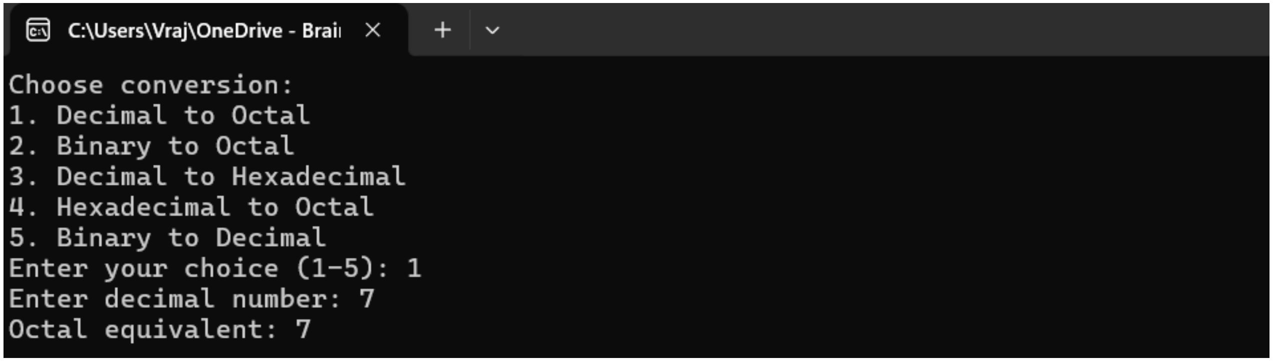
3. Decimal to Hexadecimal

4. Hexadecimal to Octal

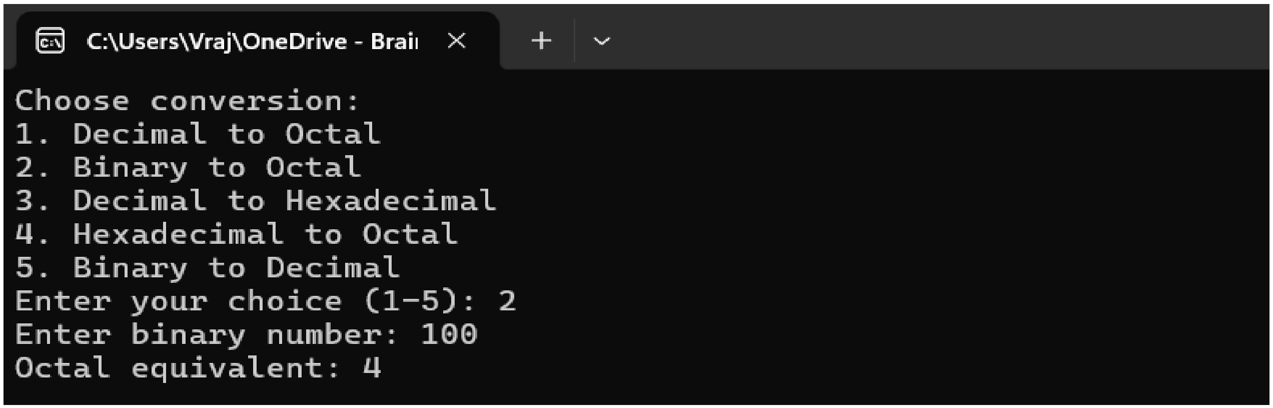
5. Binary to Decimal

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;

// 1) Decimal to Octal Conversion
int decimalToOctal(int decimal) {
    int octal = 0, i = 1;
    while (decimal != 0) {
        octal += (decimal % 8) * i;
        decimal /= 8;
        i *= 10;
    }
    return octal;
}

// 2️⃣ Binary to Octal Conversion
int binaryToDecimal(long long binary) {
    int decimal = 0, i = 0;
    while (binary != 0) {
        int digit = binary % 10;
        decimal += digit * pow(2, i);
        binary /= 10;
        i++;
    }
    return decimal;
}

int binaryToOctal(long long binary) {
    int decimal = binaryToDecimal(binary);
    return decimalToOctal(decimal);
}

// 3️⃣ Decimal to Hexadecimal Conversion
string decimalToHexadecimal(int decimal) {
    string hex = "";
    char hexChars[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};

    while (decimal != 0) {
        hex = hexChars[decimal % 16] + hex;
        decimal /= 16;
    }
    return hex;
}

// 4️⃣ Hexadecimal to Octal Conversion
int hexToDecimal(string hex) {
    int decimal = 0, base = 1;
    for (int i = hex.length() - 1; i >= 0; i--) {
        if (hex[i] >= '0' && hex[i] <= '9') {
            decimal += (hex[i] - '0') * base;
        } else {
            decimal += (hex[i] - 'A' + 10) * base;
        }
        base *= 16;
    }
    return decimal;
}

int hexToOctal(string hex) {
    int decimal = hexToDecimal(hex);
    return decimalToOctal(decimal);
}

// 5□Binary to Decimal Conversion
int binaryToDecimalConversion(long long binary) {
    return binaryToDecimal(binary);
}

int main() {
    int choice;
    cout << "Choose conversion:\n";
    cout << "1. Decimal to Octal\n";
    cout << "2. Binary to Octal\n";
    cout << "3. Decimal to Hexadecimal\n";
    cout << "4. Hexadecimal to Octal\n";
    cout << "5. Binary to Decimal\n";
    cout << "Enter your choice (1-5): ";
    cin >> choice;

    if (choice == 1) {
        int decimal;
        cout << "Enter decimal number: ";
        cin >> decimal;
        cout << "Octal equivalent: " << decimalToOctal(decimal) << endl;
    }
    else if (choice == 2) {
        long long binary;
        cout << "Enter binary number: ";
        cin >> binary;
        cout << "Octal equivalent: " << binaryToOctal(binary) << endl;
    }
    else if (choice == 3) {
        int decimal;
        cout << "Enter decimal number: ";
        cin >> decimal;
        cout << "Hexadecimal equivalent: " << decimalToHexadecimal(decimal) <<
endl;
    }
    else if (choice == 4) {
        string hex;
        cout << "Enter hexadecimal number: ";
        cin >> hex;
        cout << "Octal equivalent: " << hexToOctal(hex) << endl;
    }
    else if (choice == 5) {
        long long binary;
        cout << "Enter binary number: ";
        cin >> binary;
        cout << "Decimal equivalent: " << binaryToDecimalConversion(binary) <<
endl;
    }
    else {
        cout << "Invalid choice! Please select a valid option." << endl;
    }

    return 0;
}
```

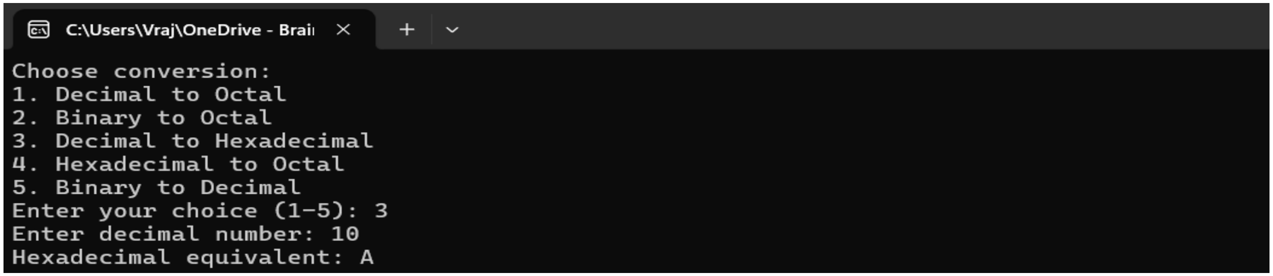
Output:



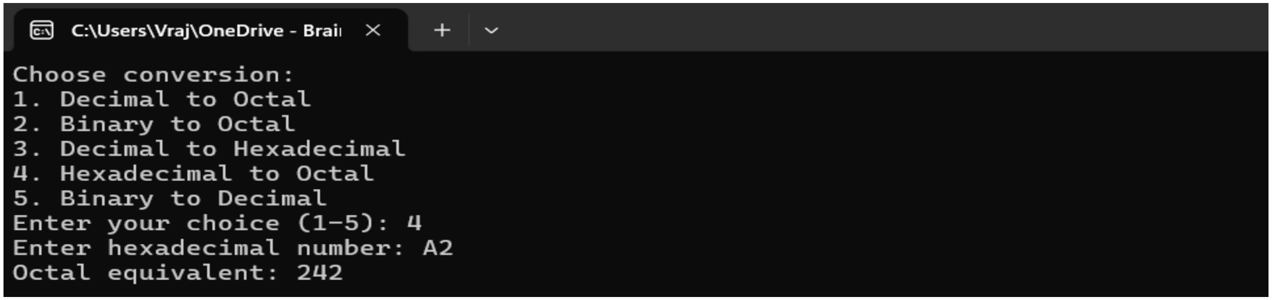
```
Choose conversion:
1. Decimal to Octal
2. Binary to Octal
3. Decimal to Hexadecimal
4. Hexadecimal to Octal
5. Binary to Decimal
Enter your choice (1-5): 1
Enter decimal number: 7
Octal equivalent: 7
```



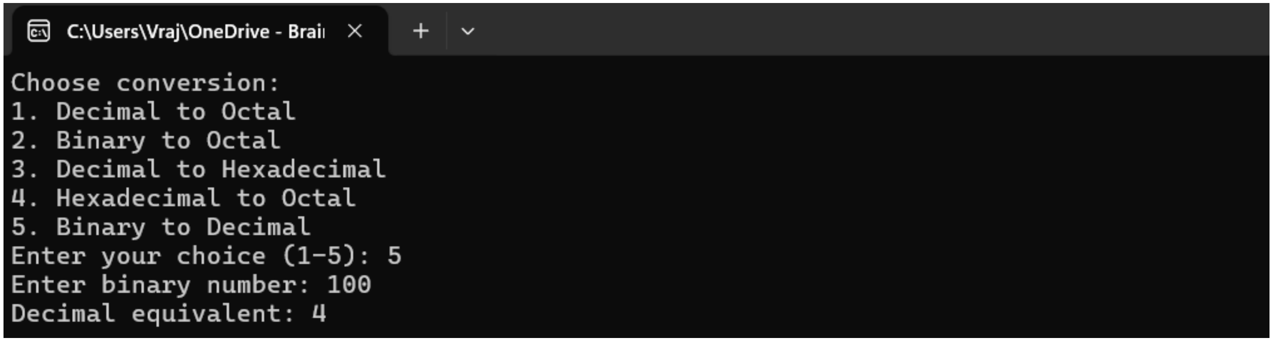
```
Choose conversion:
1. Decimal to Octal
2. Binary to Octal
3. Decimal to Hexadecimal
4. Hexadecimal to Octal
5. Binary to Decimal
Enter your choice (1-5): 2
Enter binary number: 100
Octal equivalent: 4
```



```
Choose conversion:
1. Decimal to Octal
2. Binary to Octal
3. Decimal to Hexadecimal
4. Hexadecimal to Octal
5. Binary to Decimal
Enter your choice (1-5): 3
Enter decimal number: 10
Hexadecimal equivalent: A
```



```
Choose conversion:
1. Decimal to Octal
2. Binary to Octal
3. Decimal to Hexadecimal
4. Hexadecimal to Octal
5. Binary to Decimal
Enter your choice (1-5): 4
Enter hexadecimal number: A2
Octal equivalent: 242
```



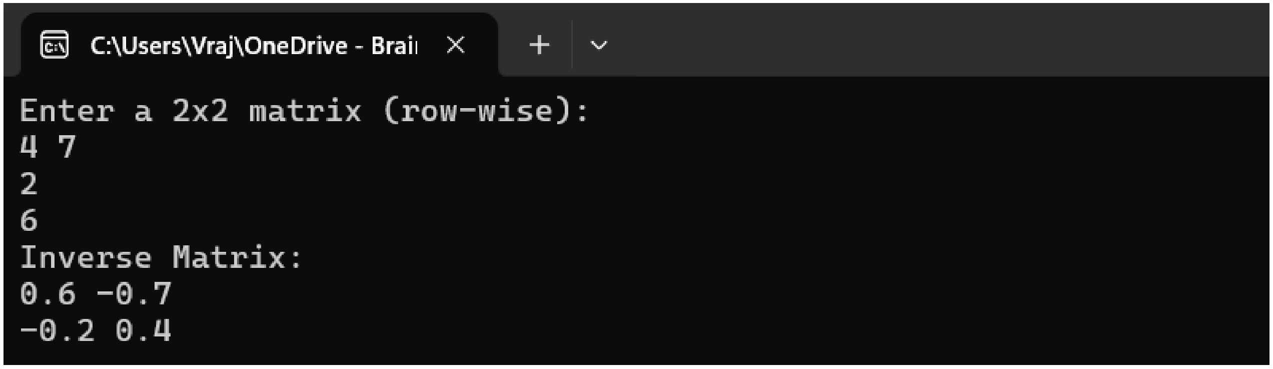
```
Choose conversion:
1. Decimal to Octal
2. Binary to Octal
3. Decimal to Hexadecimal
4. Hexadecimal to Octal
5. Binary to Decimal
Enter your choice (1-5): 5
Enter binary number: 100
Decimal equivalent: 4
```

Practical-4: Inverse Of Matrix (2*2):

```
#include <iostream>

using namespace std;

void inverseMatrix(float matrix[2][2]) {
   float determinant = matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];

   if (determinant == 0) {
      cout << "Matrix is not invertible!" << endl;
      return;
   }

   float inverse[2][2];
   inverse[0][0] = matrix[1][1] / determinant;
   inverse[0][1] = -matrix[0][1] / determinant;
   inverse[1][0] = -matrix[1][0] / determinant;
   inverse[1][1] = matrix[0][0] / determinant;

   cout << "Inverse Matrix:" << endl;
   for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
         cout << inverse[i][j] << " ";
      }
      cout << endl;
   }
}

int main() {
   float matrix[2][2];

   cout << "Enter a 2x2 matrix (row-wise): ";
   for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
         cin >> matrix[i][j];
```

```
        }
    }

    inverseMatrix(matrix);

    return 0;
}
```

Output:

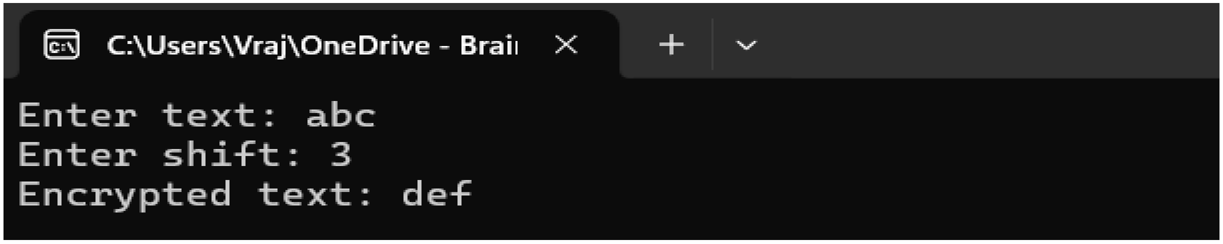


```
Enter a 2x2 matrix (row-wise):
4 7
2
6
Inverse Matrix:
0.6 -0.7
-0.2 0.4
```

Practical – 5 : Caesar Cipher Encryption

```
#include <iostream>
#include <string>
using namespace std;

string caesarCipher(string text, int shift) {
    string result = "";
    for (int i = 0; i < text.length(); i++) {
        char ch = text[i];
        if (isalpha(ch)) {
            char base = isupper(ch) ? 'A' : 'a';
            result += char(int(base + (ch - base + shift) % 26));
        } else {
            result += ch;
        }
    }
    return result;
}

int main() {
    string text;
    int shift;
    cout << "Enter text: ";
    getline(cin, text);
    cout << "Enter shift: ";
    cin >> shift;
    cout << "Encrypted text: " << caesarCipher(text, shift) << endl;
    return 0;
}
```

Output:

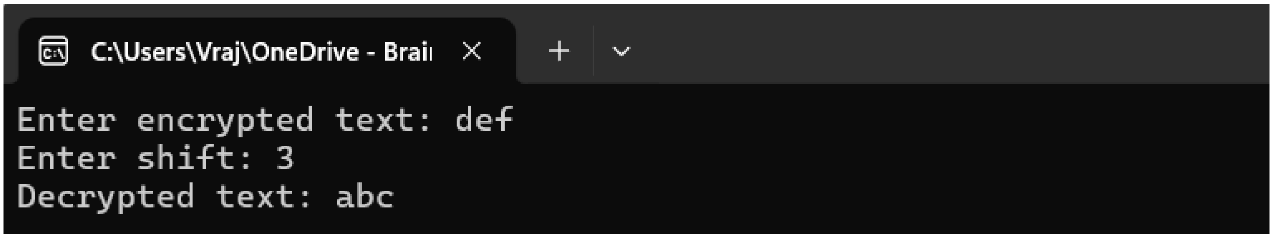


```
Enter text: abc
Enter shift: 3
Encrypted text: def
```

Caeser Cipher Decryption:

```
#include <iostream>
#include <string>
using namespace std;

string caesarDecipher(string text, int shift) {
    string result = "";
    for (int i = 0; i < text.length(); i++) {
        char ch = text[i];
        if (isalpha(ch)) {
            char base = isupper(ch) ? 'A' : 'a';
            result += char(int(base + (ch - base - shift + 26) % 26));
        } else {
            result += ch;
        }
    }
    return result;
}
int main() {
    string text;
    int shift;
    cout << "Enter encrypted text: ";
    getline(cin, text);
    cout << "Enter shift: ";
    cin >> shift;
    cout << "Decrypted text: " << caesarDecipher(text, shift) << endl;
    return 0;
}
```

Output:

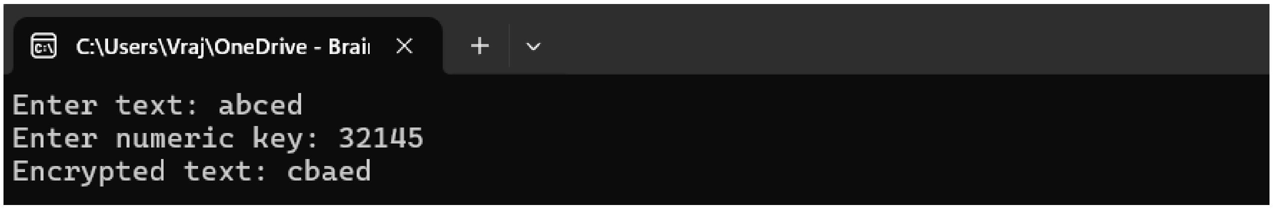


```
Enter encrypted text: def
Enter shift: 3
Decrypted text: abc
```

Practical-6: Row Transposition Cipher Encryption:

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

string rowTranspositionEncrypt(string text, string key) {
    int columns = key.length();
    int rows = (text.length() + columns - 1) / columns;

    vector< vector<char> > grid(rows, vector<char>(columns, 'X'));

    int index = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            if (index < text.length()) {
                grid[i][j] = text[index++];
            }
        }
    }

    vector< pair<char, int> > keyOrder;
    for (int i = 0; i < columns; i++) {
        keyOrder.push_back(make_pair(key[i], i));
    }
    sort(keyOrder.begin(), keyOrder.end());

    string encryptedText = "";
    for (int i = 0; i < columns; i++) {
        int col = keyOrder[i].second;
        for (int j = 0; j < rows; j++) {
            encryptedText += grid[j][col];
        }
    }
    return encryptedText;
}

int main() {
    string text, key;
    cout << "Enter text: ";
    getline(cin, text);
    cout << "Enter numeric key: ";
    cin >> key;

    string encrypted = rowTranspositionEncrypt(text, key);
    cout << "Encrypted text: " << encrypted << endl;

    return 0;
}
```

Output:

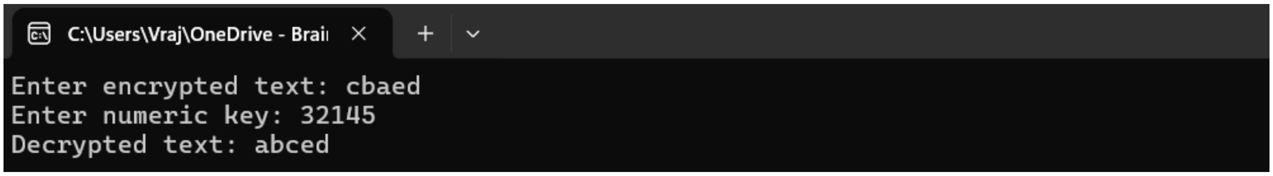


```
Enter text: abced
Enter numeric key: 32145
Encrypted text: cbaed
```

Raw Transposition Cipher Decryption:

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

string rowTranspositionDecrypt(string cipher, string key) {
    int columns = key.length();
    int rows = (cipher.length() + columns - 1) / columns;

    vector< vector<char> > grid(rows, vector<char>(columns, ' '));

    vector< pair<char, int> > keyOrder;
    for (int i = 0; i < columns; i++) {
        keyOrder.push_back(make_pair(key[i], i));
    }
    sort(keyOrder.begin(), keyOrder.end());

    int index = 0;
    for (int i = 0; i < columns; i++) {
        int col = keyOrder[i].second;
        for (int j = 0; j < rows; j++) {
            if (index < cipher.length()) {
                grid[j][col] = cipher[index++];
            }
        }
    }

    string decryptedText = "";
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            if (grid[i][j] != 'X') {
                decryptedText += grid[i][j];
            }
        }
    }
    return decryptedText;
}

int main() {
    string cipher, key;
    cout << "Enter encrypted text: ";
    getline(cin, cipher);
    cout << "Enter numeric key: ";
    cin >> key;

    string decrypted = rowTranspositionDecrypt(cipher, key);
    cout << "Decrypted text: " << decrypted << endl;

    return 0;
}
```

Output:



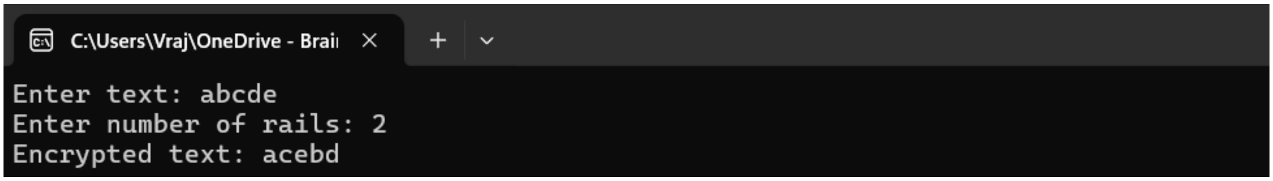
```
Enter encrypted text: cbaed
Enter numeric key: 32145
Decrypted text: abced
```

Practical-7: Rail fence Cipher encryption:

```
#include <iostream>
```

```
#include <vector>

using namespace std;

string railFenceEncrypt(string text, int rails) {
    if (rails <= 1) return text;

    vector<string> rail(rails);
    int row = 0, direction = 1;

    for (int i = 0; i < text.length(); i++) {
        rail[row] += text[i];
        row += direction;
        if (row == 0 || row == rails - 1) {
            direction = -direction;
        }
    }

    string encryptedText = "";
    for (int i = 0; i < rails; i++) {
        encryptedText += rail[i];
    }
    return encryptedText;
}

int main() {
    string text;
    int rails;

    cout << "Enter text: ";
    getline(cin, text);
    cout << "Enter number of rails: ";
    cin >> rails;

    string encrypted = railFenceEncrypt(text, rails);
    cout << "Encrypted text: " << encrypted << endl;

    return 0;
}
```

Output:

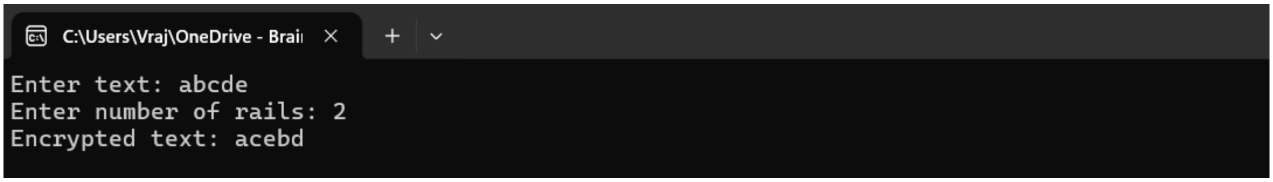


```
Enter text: abcde
Enter number of rails: 2
Encrypted text: acebd
```

Rail Fence Decryption:

```
#include <iostream>
#include <vector>

using namespace std;

string railFenceDecrypt(string cipher, int rails) {
    if (rails <= 1) return cipher;

    vector<vector<char> > rail(rails, vector<char>(cipher.length(), '\n'));
    int row = 0, direction = 1;

    for (int i = 0; i < cipher.length(); i++) {
        rail[row][i] = '*';
        row += direction;
        if (row == 0 || row == rails - 1) {
            direction = -direction;
        }
    }

    int index = 0;
    for (int i = 0; i < rails; i++) {
        for (int j = 0; j < cipher.length(); j++) {
            if (rail[i][j] == '*' && index < cipher.length()) {
                rail[i][j] = cipher[index++];
            }
        }
    }

    string decryptedText = "";
    row = 0, direction = 1;
    for (int i = 0; i < cipher.length(); i++) {
        decryptedText += rail[row][i];
        row += direction;
        if (row == 0 || row == rails - 1) {
            direction = -direction;
        }
    }

    return decryptedText;
}

int main() {
    string cipher;
    int rails;

    cout << "Enter encrypted text: ";
    cin >> cipher;
    cout << "Enter number of rails: ";
    cin >> rails;

    string decrypted = railFenceDecrypt(cipher, rails);
    cout << "Decrypted text: " << decrypted << endl;

    return 0;
}
```

Output:

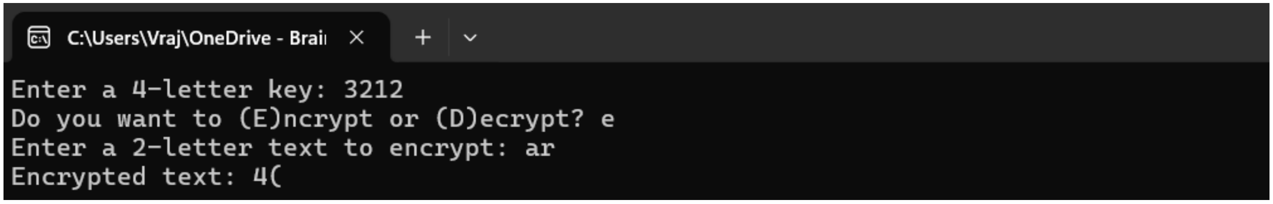


```
Enter text: abcde
Enter number of rails: 2
Encrypted text: acebd
```

Practical-8: Hill Cipher 2*2 Encryption & Decryption :

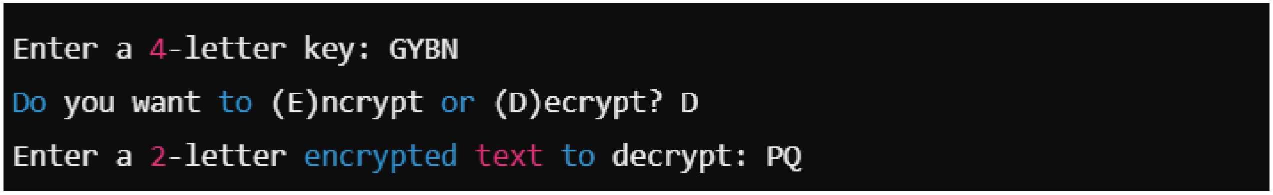
```
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int> > getKeyMatrix(string key) {
    vector<vector<int> > keyMatrix(2, vector<int>(2));
    int k = 0;
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            keyMatrix[i][j] = key[k++] - 'A';
        }
    }
    return keyMatrix;
}

vector<int> getTextVector(string text) {
    vector<int> textVector(2);
    for (int i = 0; i < 2; i++) {
        textVector[i] = text[i] - 'A';
    }
    return textVector;
}

string hillCipherEncrypt(string text, vector<vector<int> > keyMatrix) {
    vector<int> textVector = getTextVector(text);
    vector<int> encryptedVector(2);

    for (int i = 0; i < 2; i++) {
        encryptedVector[i] = (keyMatrix[i][0] * textVector[0] + keyMatrix[i][1] *
textVector[1]) % 26;
    }

    string encryptedText = "";
    for (int i = 0; i < 2; i++) {
        encryptedText += char(encryptedVector[i] + 'A');
    }
    return encryptedText;
}

int modInverse(int a, int m) {
    for (int x = 1; x < m; x++) {
        if ((a * x) % m == 1) {
            return x;
        }
    }
    return -1;
}

vector<vector<int> > getInverseKeyMatrix(vector<vector<int> > keyMatrix) {
    int det = (keyMatrix[0][0] * keyMatrix[1][1] - keyMatrix[0][1] *
keyMatrix[1][0]) % 26;
    if (det < 0) det += 26;

    int detInverse = modInverse(det, 26);
    if (detInverse == -1) {
        cout << "Key matrix is not invertible! Choose another key.\n";
        exit(0);
    }

    vector<vector<int> > inverseKeyMatrix(2, vector<int>(2));
    inverseKeyMatrix[0][0] = (keyMatrix[1][1] * detInverse) % 26;
    inverseKeyMatrix[0][1] = (-keyMatrix[0][1] * detInverse + 26) % 26;
    inverseKeyMatrix[1][0] = (-keyMatrix[1][0] * detInverse + 26) % 26;
    inverseKeyMatrix[1][1] = (keyMatrix[0][0] * detInverse) % 26;
    return inverseKeyMatrix;
}

string hillCipherDecrypt(string cipher, vector<vector<int> > keyMatrix) {
    vector<vector<int> > inverseKeyMatrix = getInverseKeyMatrix(keyMatrix);
    vector<int> cipherVector = getTextVector(cipher);
    vector<int> decryptedVector(2);

    for (int i = 0; i < 2; i++) {
        decryptedVector[i] = (inverseKeyMatrix[i][0] * cipherVector[0] +
inverseKeyMatrix[i][1] * cipherVector[1]) % 26;
    }

    string decryptedText = "";
    for (int i = 0; i < 2; i++) {
        decryptedText += char(decryptedVector[i] + 'A');
    }
    return decryptedText;
}

int main() {
    string key, text, choice;

    cout << "Enter a 4-letter key: ";
    cin >> key;
    cout << "Do you want to (E)ncrypt or (D)ecrypt? ";
    cin >> choice;

    if (choice == "E" || choice == "e") {
        cout << "Enter a 2-letter text to encrypt: ";
        cin >> text;
        vector<vector<int> > keyMatrix = getKeyMatrix(key);
        string encrypted = hillCipherEncrypt(text, keyMatrix);
        cout << "Encrypted text: " << encrypted << endl;
    }
    else if (choice == "D" || choice == "d") {
        cout << "Enter a 2-letter encrypted text to decrypt: ";
        cin >> text;
        vector<vector<int> > keyMatrix = getKeyMatrix(key);
        string decrypted = hillCipherDecrypt(text, keyMatrix);
        cout << "Decrypted text: " << decrypted << endl;
    }
    else {
        cout << "Invalid choice! Please enter 'E' for encryption or 'D' for decryption."
<< endl;
    }
    return 0;
}
```

Output For Encryption :

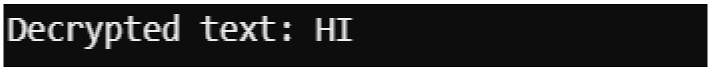


```
Enter a 4-letter key: 3212
Do you want to (E)ncrypt or (D)ecrypt? e
Enter a 2-letter text to encrypt: ar
Encrypted text: 4(
```

Output For Decryption:



```
Enter a 4-letter key: GYBN
Do you want to (E)ncrypt or (D)ecrypt? D
Enter a 2-letter encrypted text to decrypt: PQ
```



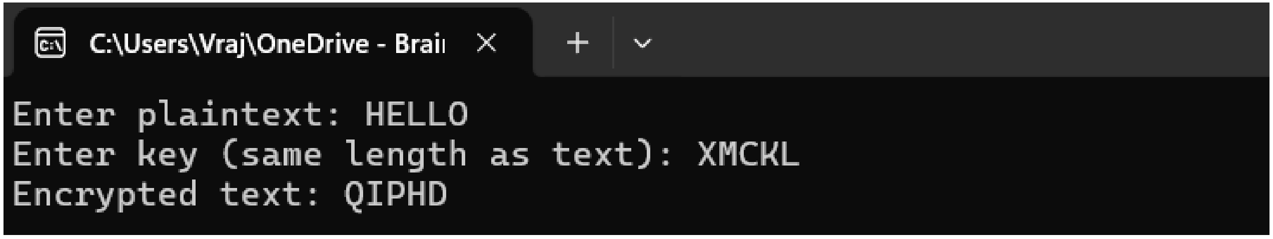
```
Decrypted text: HI
```

Practical-9: Vernam Cipher Encryption:

```
#include <iostream>
using namespace std;

string vernamEncrypt(string text, string key) {
    string encryptedText = "";
    for (int i = 0; i < text.length(); i++) {
        encryptedText += char((text[i] ^ key[i]) + 'A');
    }
    return encryptedText;
}
```

```
int main() {
    string text, key;
    cout << "Enter plaintext: ";
    cin >> text;
    cout << "Enter key (same length as text): ";
    cin >> key;

    if (text.length() != key.length()) {
        cout << "Error: Key must be the same length as plaintext!" << endl;
        return 1;
    }

    string encrypted = vernamEncrypt(text, key);
    cout << "Encrypted text: " << encrypted << endl;

    return 0;
}
```

Output:

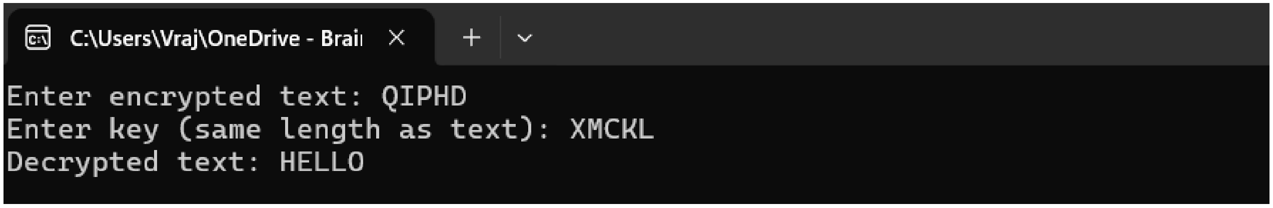


```
Enter plaintext: HELLO
Enter key (same length as text): XMCKL
Encrypted text: QIPHD
```

Vernam Cipher Decryption:

```
#include <iostream>
using namespace std;

string vernamDecrypt(string cipher, string key) {
    string decryptedText = "";
    for (int i = 0; i < cipher.length(); i++) {
        decryptedText += char((cipher[i] - 'A') ^ key[i]);
    }
    return decryptedText;
}

int main() {
    string cipher, key;
    cout << "Enter encrypted text: ";
    cin >> cipher;
    cout << "Enter key (same length as text): ";
    cin >> key;

    if (cipher.length() != key.length()) {
        cout << "Error: Key must be the same length as encrypted text!" << endl;
        return 1;
    }

    string decrypted = vernamDecrypt(cipher, key);
    cout << "Decrypted text: " << decrypted << endl;

    return 0;
}
```

Output:

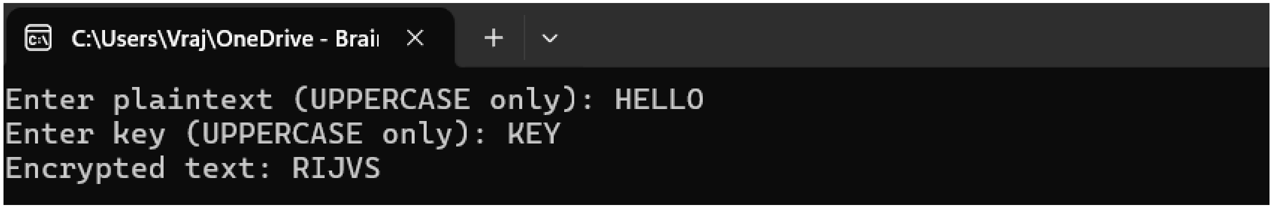


```
Enter encrypted text: QIPHD
Enter key (same length as text): XMCKL
Decrypted text: HELLO
```

Practical-10: Polyalphabetic Cipher Encryption:

```
#include <iostream>
using namespace std;

string generateKey(string text, string key) {
    while (key.length() < text.length()) {
        key += key;
    }
    return key.substr(0, text.length());
}

string vigenereEncrypt(string text, string key) {
    string encryptedText = "";
    key = generateKey(text, key);

    for (int i = 0; i < text.length(); i++) {
        char encryptedChar = ((text[i] - 'A') + (key[i] - 'A')) % 26 + 'A';
        encryptedText += encryptedChar;
    }
    return encryptedText;
}

int main() {
    string text, key;
    cout << "Enter plaintext (UPPERCASE only): ";
    cin >> text;
    cout << "Enter key (UPPERCASE only): ";
    cin >> key;

    string encrypted = vigenereEncrypt(text, key);
    cout << "Encrypted text: " << encrypted << endl;

    return 0;
}
```

Output:



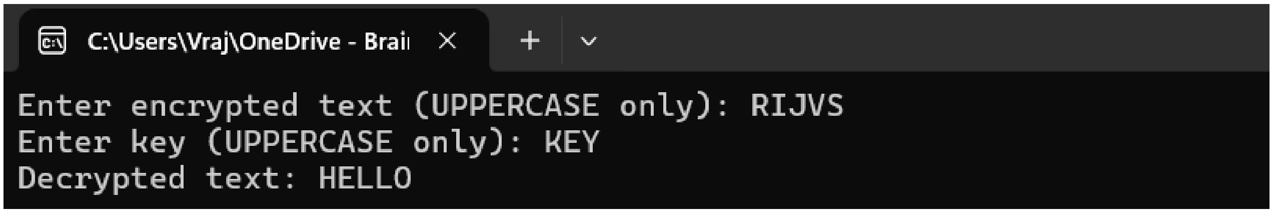
```
Enter plaintext (UPPERCASE only): HELLO
Enter key (UPPERCASE only): KEY
Encrypted text: RIJVS
```

Polyalphabetic Cipher Decryption:

```
#include <iostream>
```

```
using namespace std;

string generateKey(string text, string key) {
    while (key.length() < text.length()) {
        key += key;
    }
    return key.substr(0, text.length());
}

string vigenereDecrypt(string cipher, string key) {
    string decryptedText = "";
    key = generateKey(cipher, key);

    for (int i = 0; i < cipher.length(); i++) {
        char decryptedChar = ((cipher[i] - key[i] + 26) % 26) + 'A';
        decryptedText += decryptedChar;
    }
    return decryptedText;
}

int main() {
    string cipher, key;
    cout << "Enter encrypted text (UPPERCASE only): ";
    cin >> cipher;
    cout << "Enter key (UPPERCASE only): ";
    cin >> key;

    string decrypted = vigenereDecrypt(cipher, key);
    cout << "Decrypted text: " << decrypted << endl;

    return 0;
}
```

Output:



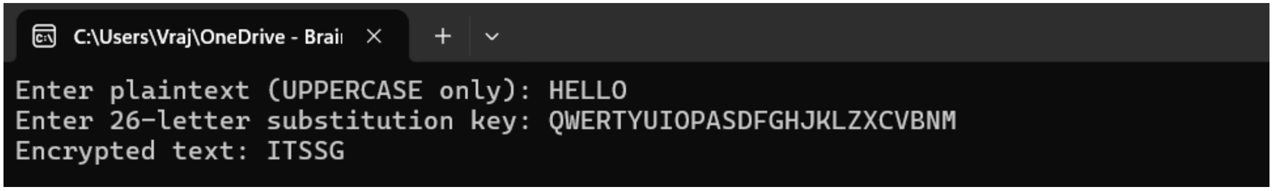
```
Enter encrypted text (UPPERCASE only): RIJVS
Enter key (UPPERCASE only): KEY
Decrypted text: HELLO
```

Practical-11: Mono-alphabetic Cipher Encryption:

```
#include <iostream>
using namespace std;

string monoalphabeticEncrypt(string text, string key) {
    string encryptedText = "";
    string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    for (int i = 0; i < text.length(); i++) {
        if (isalpha(text[i])) {
            char upperChar = toupper(text[i]);
            int index = alphabet.find(upperChar);
            encryptedText += key[index];
        } else {
            encryptedText += text[i];
        }
    }
    return encryptedText;
}

int main() {
    string text, key;
    cout << "Enter plaintext (UPPERCASE only): ";
    cin >> text;
    cout << "Enter 26-letter substitution key: ";
    cin >> key;

    string encrypted = monoalphabeticEncrypt(text, key);
    cout << "Encrypted text: " << encrypted << endl;
```

```
  return 0;
}
```

Output:



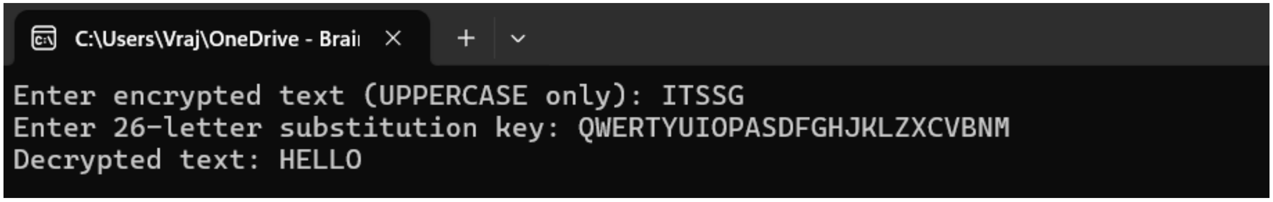
```
Enter plaintext (UPPERCASE only): HELLO
Enter 26-letter substitution key: QWERTYUIOPASDFGHJKLZXCVBNM
Encrypted text: ITSSG
```

Mono-Alphabetic Decryption:

```
#include <iostream>
using namespace std;

string monoalphabeticDecrypt(string cipher, string key) {
   string decryptedText = "";
   string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

   for (int i = 0; i < cipher.length(); i++) {
      if (isalpha(cipher[i])) {
         char upperChar = toupper(cipher[i]);
         int index = key.find(upperChar);
         decryptedText += alphabet[index];
      } else {
         decryptedText += cipher[i];
      }
   }
   return decryptedText;
}

int main() {
   string cipher, key;
   cout << "Enter encrypted text (UPPERCASE only): ";
   cin >> cipher;
   cout << "Enter 26-letter substitution key: ";
   cin >> key;
```

```
    string decrypted = monoalphabeticDecrypt(cipher, key);
    cout << "Decrypted text: " << decrypted << endl;

    return 0;
}
```

Output:



```
Enter encrypted text (UPPERCASE only): ITSSG
Enter 26-letter substitution key: QWERTYUIOPASDFGHJKLZXCVBNM
Decrypted text: HELLO
```