Name : Vishal Madle
PRN : 2020BTECS00092
Batch : B8

CNS lab

Experiment 01

Aim: To implement Caesar cipher

Theory: The Caesar cipher is a simple and widely known substitution cipher technique used for encryption. It was named after Julius Caesar, who is historically believed to have used this method to encode his private correspondence. The Caesar cipher is a basic encryption technique where each letter in a message is shifted a fixed number of positions down or up the alphabet.

Code:

```cpp
#include <bits/stdc++.h>

using namespace std;

char shift_char(char c,int shift,char
    op){if(isalpha(c) && op=='e'){
        char base=islower(c) ? 'a' : 'A';
        return char((c - base + shift) % 26 + base);
    }
    else if(isalpha(c)  &&
        op=='d'){ char base=islower(c) ?
        'a': 'A';
        return char((c - base - shift + 26) % 26 + base);
    }
    return c;
}

string encrypt_text(string text,int
    key){string encrypted="";
    for(char
        c:text){ encrypted+=shift_char(c,k
        ey,'e');
    }
    return encrypted;
}

string decrypt_text(string text,int
    key){string decrypted="";
    for(char
        c:text){ decrypted+=shift_char(c,k
        ey,'d');
```

```
int main(int argc, char const *argv[])
{
    int choice,key;
    string text;
    cout<<"Enter choice: ";
    cout<<endl<<"1. Encrypt | 2. Decrypt"<<endl;
    cin>>choice;
    cin.get();
    if(choice==1){
        cout<<"enter text: ";
        getline(cin,text);
        cout<<"Enter key: ";
        cin>>key;
        string result=encrypt_text(text,key);
        cout<<"encrypted text: "<<result<<endl;
    }
    else if(choice==2){
        cout<<"enter encrypted text: ";
        getline(cin,text);
        cout<<"Enter key: ";
        cin>>key;
        string result=decrypt_text(text,key);
        cout<<"decrypted text: "<<result<<endl;
    }
    else{
        cout<<"not a valid choice!"<<endl;
    }
    return 0;
}
```

Output:

```
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp1> c
pher } ; if ($?) { .\caesar_cipher }
Enter choice:
1. Encrypt | 2. Decrypt
1
enter text: meetmeintheparktodaynoon
Enter key: 4
encrypted text: qiixqimrxlitevoxshecrssr
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp1> c
pher } ; if ($?) { .\caesar_cipher }
Enter choice:
1. Encrypt | 2. Decrypt
2
enter encrypted text: qiixqimrxlitevoxshecrssr
Enter key: 4
decrypted text: meetmeintheparktodaynoon
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp1>
```

Name : Vishal Madle
PRN : 2020BTECS00092
Batch : B8

CNS lab

Experiment 02

Aim: To implement Columnar cipher

Theory: The Columnar Cipher is a transposition cipher where the plaintext is written out in rows and then read out again column by column in sorted order of characters in key. The key determines the number of columns.

Code:

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

string alpha_lower(string
    text){for(char c:text){
        if(isalnum(c)){ c
            =tolower(c);
        }
    }
    return text;
}

string encrypt(string text, string key)
    {map<char, vector<char>> mp;
    int cnt = 0;
    for (int i = 0; i < text.size(); i++)
        {if (cnt == key.size()) cnt = 0;

        mp[key[cnt++]].push_back(text[i]);
    }

    string encrypted;
    for (auto i : mp) {
        for (auto j: i.second)
            {encrypted+=j;
        }
    }

    return encrypted;
}

string decrypt(string cipher, string key) {
```

```
    map<int, int> map1;

    int common = cipher.size() / key.size();
    int extra = cipher.size() % key.size();

    for (int i = 0; i < key.size(); i++)
        {if (i < extra)
            map1[i] = common + 1;
        else
            map1[i] = common;
    }

    map<int, vector<char>> map2;

    int start = 0;

    string sortedKey=key;
    sort(sortedKey.begin(), sortedKey.end());

    for (int i = 0; i < sortedKey.size(); i++)
        {for (int j = 0; j < key.size(); j++) {
            if (sortedKey[i] == key[j]) {
                for (int k = 0; k < map1[j]; k++)
                    { map2[key[j]].push_back(cipher[start++]);
                }
            }
        }
    }

    string plain;

    vector<int> counters(key.size(), 0);

    while (plain.size() < cipher.size()) {
        for (int i = 0; i < key.size(); i++)
            {if (counters[i] < map1[i])
                plain += map2[key[i]][counters[i]++];
        }
    }
    return plain;
}

int main() {
    int choice;
    cout<<"Enter choice: ";
    cout<<endl<<"1. Encrypt | 2. Decrypt"<<endl;
    cin>>choice;
    cin.get();
```

```
    if (choice == 1)
        { string text,
        key;
        cout << "\nEnter text: ";
        getline(cin, text);
        text = alpha_lower(text);

        cout << "\nEnter key: ";
        getline(cin, key);
        alpha_lower(key);

        string cipher = encrypt(text, key);

        cout << "\nEncrypted text is : " << cipher << endl;
    } else if (choice == 2)
        {string cipher, key;
        cout << "\nEnter cipher text: ";
        getline(cin, cipher);
        cipher = alpha_lower(cipher);

        cout << "\nEnter key: ";
        getline(cin, key);
        alpha_lower(key);

        string text = decrypt(cipher, key);

        cout << "\nDecrypted text is : " << text << endl;
    }

    return 0;
}
```

Output:

```
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp2> cd
 -o columnar_cipher } ; if ($?) { .\columnar_cipher }
Enter choice:
1. Encrypt | 2. Decrypt
1

Enter text: killerisfromyourcolony

Enter key:  clue

Encrypted text is : kefycnlsmroirrooylioul
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp2> cd
 -o columnar_cipher } ; if ($?) { .\columnar_cipher }
Enter choice:
1. Encrypt | 2. Decrypt
2

Enter cipher text: kefycnlsmroirrooylioul

Enter key: clue

Decrypted text is : killerisfromyourcolony
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp2>
```

Name : Vishal Madle
PRN : 2020BTECS00092
Batch : B8

CNS lab

Experiment 03

Aim: To implement Playfair cipher

Theory: The Playfair Cipher is a digraph substitution cipher that encrypts pairs of letters at a time. It uses a 5x5 grid containing a keyword (omitting duplicates) for encryption. To encrypt, you find each pair of letters in the grid and apply specific rules. If the letters are in the same row, they are replaced with the letters to their right, and if in the same column, with the letters below. If neither, they form a rectangle and are replaced with the letters at the opposite corners.

Code:

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

const int SIZE = 5;

void generateMatrix(string key, char matrix[SIZE][SIZE])
    {string keyWithoutDuplicates = "";
    bool used[26] = { false };

    for(char c : key) {
        if(c != 'J' && !used[c - 'A'])
            {keyWithoutDuplicates += c;
            used[c - 'A'] = true;
        }
    }

    int index = 0;
    for(int i = 0; i < SIZE; i++)
        { for(int j = 0; j < SIZE; j++)
        {
            if(index < keyWithoutDuplicates.length())
                { matrix[i][j] =
                keyWithoutDuplicates[index++];
            } else {
                for(char c = 'A'; c <= 'Z'; c++) {
                    if(c != 'J' && !used[c - 'A'])
                        {matrix[i][j] = c;
                        used[c - 'A'] = true;
                        break;
```

```
            }
        }
    }
}

void findPosition(char matrix[SIZE][SIZE], char c, int& row, int& col)
    {if (c == 'J') c = 'I';

    for(int i = 0; i < SIZE; i++)
        { for(int j = 0; j < SIZE; j++)
        {
            if(matrix[i][j] == c)
                {row = i;
                col = j;
                return;
            }
        }
    }
}

string encryptDigraph(char matrix[SIZE][SIZE], char a, char b)
    {int rowA, colA, rowB, colB;
    findPosition(matrix, a, rowA, colA);
    findPosition(matrix, b, rowB, colB);

    if(rowA == rowB) {
        return string(1, matrix[rowA][(colA + 1) % SIZE]) + string(1,
matrix[rowB][(colB + 1) % SIZE]);
    }
    if(colA == colB) {
        return string(1, matrix[(rowA + 1) % SIZE][colA]) + string(1,
matrix[(rowB + 1) % SIZE][colB]);
    }
    return string(1, matrix[rowA][colB]) + string(1, matrix[rowB][colA]);
}

string decryptDigraph(char matrix[SIZE][SIZE], char a, char b)
    {int rowA, colA, rowB, colB;
    findPosition(matrix, a, rowA, colA);
    findPosition(matrix, b, rowB, colB);


    if(rowA == rowB) {
        return string(1, matrix[rowA][(colA - 1 + SIZE) % SIZE]) + string(1,
matrix[rowB][(colB - 1 + SIZE) % SIZE]);
    }

    if(colA == colB) {
```

```
        return string(1, matrix[(rowA - 1 + SIZE) % SIZE][colA]) + string(1,
matrix[(rowB - 1 + SIZE) % SIZE][colB]);
    }

    return string(1, matrix[rowA][colB]) + string(1, matrix[rowB][colA]);
}

int main() {
    string key, text;
    char matrix[SIZE][SIZE];

    cout << "Enter the key (uppercase, excluding J): ";
    cin >> key;

    generateMatrix(key, matrix);

    cout << "Enter the text (uppercase, without spaces): ";
    cin >> text;

    string result;
    char choice;

    cout << "Encrypt (E) or Decrypt (D)? ";
    cin >> choice;

    if(choice == 'E' || choice == 'e') {

        string preparedText = "";
        for(int i = 0; i < text.length(); i += 2)
            {if(i + 1 < text.length()) {
                if(text[i] == text[i + 1])
                    {preparedText += text[i];
                    preparedText += 'X';
                    i--;
                } else {
                    preparedText += text.substr(i, 2);
                }
            } else {
                preparedText += text[i];
                preparedText += 'X';
            }
        }
        for(int i = 0; i < preparedText.length(); i += 2)
            {char a = preparedText[i];
            char b = preparedText[i + 1];
            result += encryptDigraph(matrix, a, b);
        }
    }
```

```
    else if(choice == 'D' || choice == 'd')
        { for(int i = 0; i < text.length(); i += 2)
        {
            char a = text[i];
            char b = text[i + 1];
            result += decryptDigraph(matrix, a, b);
        }

        string cleanedText = "";
        for(int i = 0; i < result.length(); i++)
            {if(result[i] != 'X') {
                cleanedText += result[i];
            }
        }
        result = cleanedText;
    }
    else {
        cout << "Invalid choice. Please enter 'E' for Encrypt or 'D' for
Decrypt." << endl;
        return 1;
    }

    cout << "Result: " << result << endl;

    return 0;
}
```

Output:

```
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp3> cd "
r_cipher } ; if ($?) { .\playfair_cipher }
Enter the key (uppercase, excluding J): ALERT
Enter the text (uppercase, without spaces): BOBKILLEDALICE
Encrypt (E) or Decrypt (D)? E
Result: HVDHPCERBECPDL
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp3> cd "
r_cipher } ; if ($?) { .\playfair_cipher }
Enter the key (uppercase, excluding J): ALERT
Enter the text (uppercase, without spaces): HVDHPCERBECPDL
Encrypt (E) or Decrypt (D)? D
Result: BOBKILLEDALICE
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp3>
```

Name : Vishal Madle
PRN : 2020BTECS00092
Batch : B8

CNS lab

Experiment 04

Aim: To implement Railfence cipher

Theory: The Rail Fence Cipher is a type of transposition cipher. It works by writing the plaintext in a zigzag pattern across a specified number of lines (rails). The characters are then read off in rows to create the ciphertext. To decrypt, the process is reversed using the same number of rails to reveal the original message.

Code:

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

string format(string &str)
    {for (char c : str) {
        if (isalpha(c))
            { c+=tolower(c
            );
        }
    }
    return str;
}

string encrypt(string &plain, int key)
    {vector<vector<char>> matrix(key);

    int rowNumber = 0;
    int flag = 1;
    for (int i = 0; i < plain.size(); i++)
        { matrix[rowNumber].push_back(plain[i]);
        rowNumber += flag;
        if (rowNumber == 0)
            flag = 1;

        if (rowNumber == key - 1)
            flag = -1;
    }

    string cipher;
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < matrix[i].size(); j++)
            cipher += matrix[i][j];
```

```
    return cipher;
}

string decrypt(string &cipher, int key)
    {vector<vector<int>> matrixDecry(key);
    int rowNumber = 0;
    int flag = 1;

    int n = cipher.length();

    for (int i = 0; i < n; i++)
        { matrixDecry[rowNumber].push_back(i)
        ;rowNumber += flag;
        if (rowNumber == (key - 1))
            flag = -1;
        if (rowNumber == 0)
            flag = 1;
    }

    vector<int> mapping;
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < matrixDecry[i].size(); j++)
            mapping.push_back(matrixDecry[i][j]);
    }

    map<int, char> m;
    for (int i = 0; i < n; i++)
        m[mapping[i]] = cipher[i];

    string plain;
    for (int i = 0; i < n; i++)
        plain += m[i];

    return plain;
}

int main() {
    int choice;
    cout << "1. Encrypt\n2. Decrypt\nEnter your choice: ";
    cin >> choice;
    cin.get();

    if (choice == 1)
        {string plain;
        int key;
        cout << "\nEnter plain text: ";
        getline(cin, plain);
        plain = format(plain);
```

```
        cout << "\nEnter key: integer value: ";
        cin >> key;

        string cipher = encrypt(plain, key);

        cout << "\nEncrypted text is : " << cipher << endl;
    } else if (choice == 2)
        {string cipher;
        int key;
        cout << "\nEnter cipher text: ";
        getline(cin, cipher);
        cipher = format(cipher);

        cout << "\nEnter key: integer value: ";
        cin >> key;

        string plain = decrypt(cipher, key);

        cout << "\nDecrypted text is : " << plain << endl;
    }

    return 0;
}
```

Output:

```
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp4> c
nce_cipher } ; if ($?) { .\railfence_cipher }
1. Encrypt
2. Decrypt
Enter your choice: 1

Enter plain text: weaponhidedintheoffice

Enter key: integer value: 4

Encrypted text is : whnfeniitfiaoddhocpeee
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp4> c
nce_cipher } ; if ($?) { .\railfence_cipher }
1. Encrypt
2. Decrypt
Enter your choice: 2

Enter cipher text: whnfeniitfiaoddhocpeee

Enter key: integer value: 4

Decrypted text is : weaponhidedintheoffice
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp4>
```

Name : Vishal Madle
PRN : 2020BTECS00092
Batch : B8

CNS lab

Experiment 05

Aim: To implement Vigenere cipher

Theory: The Vigenère Cipher is a polyalphabetic substitution cipher that uses a keyword to shift letters in the plaintext. The keyword is repeated to match the length of the message, and each letter of the keyword determines the shift value.

Code:

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

string alpah_lower(string str)
    {for (char c : str) {
        if (isalpha(c))
            { c+=tolower(c
            );
        }
    }
    return str;
}

string encrypt(string text, string key)
    {string cipher;
    for (int i = 0; i < text.size(); i++) {
        int val = text[i] - 'a' + key[i % (key.size())] - 'a';
        cipher+=('a' + (val % 26));
    }
    return cipher;
}

string decrypt(string cipher, string key)
    {string text;
    for (int i = 0; i < cipher.size(); i++) {
        int val = cipher[i] - 'a' - (key[i % (key.size())] - 'a');
        text += ('a' + (val + 26) % 26);
    }
    return text;
}

int main() {
    int choice;
```

```
    cin >> choice;
    cin.get();

    if (choice == 1)
        { string plain,
        key;
        cout << "\nEnter plain text: ";
        getline(cin, plain);
        plain = alpah_lower(plain);

        cout << "\nEnter key: ";
        getline(cin, key);

        string cipher = encrypt(plain, key);

        cout << "\nEncrypted text is : " << cipher << endl;
    } else if (choice == 2)
        {string cipher, key;
        cout << "\nEnter cipher text: ";
        getline(cin, cipher);
        cipher = alpah_lower(cipher);

        cout << "\nEnter key: ";
        getline(cin, key);

        string plain = decrypt(cipher, key);

        cout << "\nDecrypted text is : " << plain << endl;
    }

    return 0;
}
```

Output:

```
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp5> cd
e_cipher } ; if ($?) { .\vigenere_cipher }
1. Encrypt
2. Decrypt
Enter your choice: 1

Enter plain text: agentonthemission

Enter key: secret

Encrypted text is : skgexhfxjvqbkwkfr
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp5> cd
e_cipher } ; if ($?) { .\vigenere_cipher }
1. Encrypt
2. Decrypt
Enter your choice: 2

Enter cipher text: skgexhfxjvqbkwkfr

Enter key: secret

Decrypted text is : agentonthemission
PS C:\Users\shree\Documents\My workspace 2\CNS lab\Exp5>
```