Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 3

Aim: Implementation of Playfair cipher

Description:

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In Playfair cipher unlike traditional cipher, we encrypt a pair of alphabets(digraphs) instead of a single alphabet.
It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment.

Encryption Technique

For the encryption process let us consider the following example:

Key: monarchy
Plaintext: instruments

Encryption Technique

The Playfair Cipher Encryption Algorithm:
The Algorithm consists of 2 steps:

Generate the key Square(5×5):
The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.

The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

Roll No: 2103163
Batch: C32
Name: Om Shete
Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.
For example

PlainText: "instruments"
After Split: 'in' 'st' 'ru' 'me' 'nt' 'sz'

1. A pair cannot be made with the same letter. Break the letter in a single and add a bogus letter to the previous letter.

Plain Text: “hello”
After Split: ‘he’ ‘lx’ ‘lo’
Here ‘x’ is the bogus letter.

2. If the letter is standing alone in the process of pairing, then add an extra bogus letter with the alone letter

Plain Text: “helloe”
AfterSplit: ‘he’ ‘lx’ ‘lo’ ‘ez’
Here ‘z’ is the bogus letter.

Rules for Encryption:

If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom).

For example
Diagraph: "me"
Encrypted Text: cl
Encryption:
m -> c
e -> l

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Roll No: 2103163
Batch: C32
Name: Om Shete

Rules for Encryption 1

If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position).

For example
Diagraph: "st"
Encrypted Text: tl
Encryption:
s -> t
t -> l

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Rules for Encryption 2

If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

For example
Diagraph: "nt"
Encrypted Text: rq
Encryption:
n -> r
t -> q

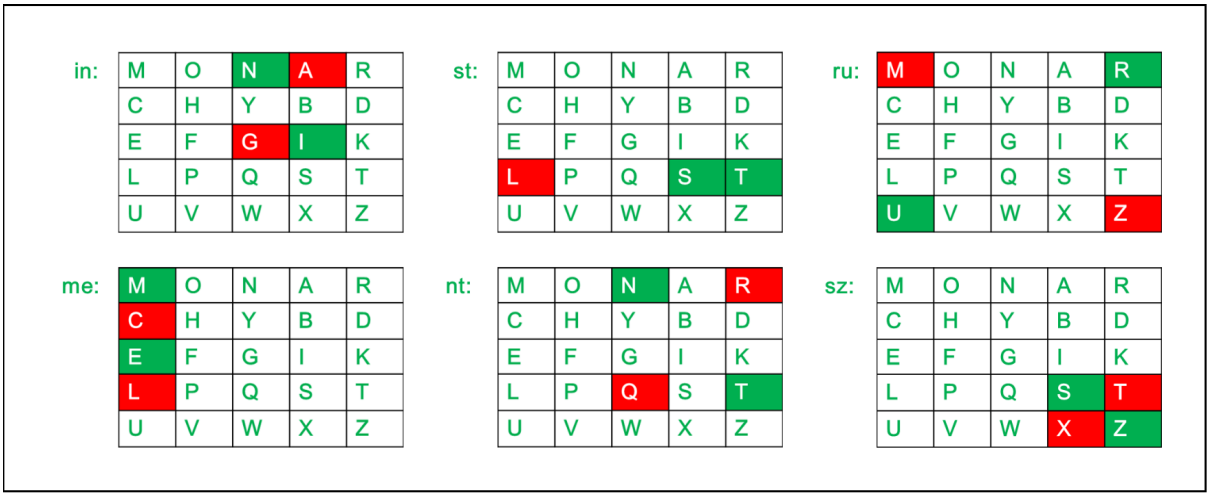
Roll No: 2103163
Batch: C32
Name: Om Shete

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Rules for Encryption 3

For example

Plain Text: "instrumentsz"
Encrypted Text: gatlmzclrqtx
Encryption:
i -> g
n -> a
s -> t
t -> l
r -> m
u -> z
m -> c
e -> l
n -> r
t -> q
s -> t
z -> x



Roll No: 2103163
Batch: C32
Name: Om Shete

Code:

```
#include <algorithm>
#include <cctype>
#include <cstring>
#include <iostream>
using namespace std;

char mat[5][5];
const char PADD_CHAR = 'x';
bool is_odd = false;

void generate_the_matrix(string key) {
  int checkSmall[26] = {0};
  int r = 0, c = 0;

  for (int i = 0; i < key.length(); i++) {
    if (key[i] == 'j') {
      key = key.substr(0, i) + 'i' + key.substr(i + 1);
    }

    if (checkSmall[key[i] - 'a'] == 0) {
      mat[r][c++] = key[i];
      checkSmall[key[i] - 'a'] = 1;
    }

    if (c == 5) {
      c = 0;
      r++;
    }
  }

  for (char ch = 'a'; ch <= 'z'; ch++) {
    if (ch == 'j') {
      continue;
    }

    if (checkSmall[ch - 'a'] == 0) {
      checkSmall[ch - 'a'] = 1;
      mat[r][c++] = ch;
    }
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
    if (c == 5) {
      c = 0;
      r++;
    }
  }
}

string partition(string text) {
  for (int i = 0; i < text.length(); i += 2) {
    if (text[i] == text[i + 1]) {
      text.insert(i + 1, 1, PADD_CHAR);
    }
  }

  if (text.length() % 2 == 1) {
    text += PADD_CHAR;
    is_odd = true;
  }

  return text;
}

string encryption(string text, int flag) {
  string result;
  cout << "Encryption Process:\n";
  for (int i = 0; i < text.length(); i += 2) {
    int p1[2], p2[2];
    p1[0] = -1;
    p1[1] = -1;
    p2[0] = -1;
    p2[1] = -1;

    for (int j = 0; j < 5; j++) {
      for (int k = 0; k < 5; k++) {
        if (mat[j][k] == text[i]) {
          p1[0] = j;
          p1[1] = k;
        }

        if (mat[j][k] == text[i + 1]) {
          p2[0] = j;
          p2[1] = k;
        }
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
        if (p1[0] != -1 && p2[0] != -1) {
          break;
        }
      }

      if (p1[0] != -1 && p2[0] != -1) {
        break;
      }
    }

    char ch1, ch2;
    if (p1[0] == p2[0]) {
      ch1 = mat[p1[0]][(p1[1] + flag) % 5];
      ch2 = mat[p2[0]][(p2[1] + flag) % 5];
    } else if (p1[1] == p2[1]) {
      ch1 = mat[(p1[0] + flag) % 5][p1[1]];
      ch2 = mat[(p2[0] + flag) % 5][p2[1]];
    } else {
      ch1 = mat[p1[0]][p2[1]];
      ch2 = mat[p2[0]][p1[1]];
    }

    result.push_back(ch1);
    result.push_back(ch2);

    cout << text[i] << text[i + 1] << " ---- " << ch1 << ch2 << "\n";
  }

  return result;
}

int main() {
  string text, key;

  cout << "Enter the text: ";
  getline(cin, text);

  cout << "Enter the Key: ";
  getline(cin, key);

  text.erase(remove_if(text.begin(), text.end(), ::isspace), text.end());
  transform(text.begin(), text.end(), text.begin(), ::tolower);
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
  key.erase(remove_if(key.begin(), key.end(), ::isspace), key.end());
  transform(key.begin(), key.end(), key.begin(), ::tolower);

  generate_the_matrix(key);

  cout << "The Playfair Matrix: " << endl;
  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
      cout << mat[i][j] << " ";
    }
    cout << endl;
  }

  string f_msg = partition(text);
  cout << "\nFormatted msg: " << f_msg << "\n" << endl;

  string cipher = encryption(f_msg, 1);
  cout << "\nEncrypted msg: " << cipher << "\n" << endl;

  cout << "Decryption Process:\n";
  string decryptedMsg = encryption(cipher, 4);
  cout << "\nDecrypted msg: " << decryptedMsg << endl;

  return 0;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

Results:

```
Enter the text: instruments
Enter the Key: monarchy
The Playfair Matrix:
m o n a r
c h y b d
e f g i k
l p q s t
u v w x z

Formatted msg: instrumentsx

Encryption Process:
in ---- ga
st ---- tl
ru ---- mz
me ---- cl
nt ---- rq
sx ---- xa

Encrypted msg: gatlmzclrqxa

Decryption Process:
Encryption Process:
ga ---- in
tl ---- st
mz ---- ru
cl ---- me
rq ---- nt
xa ---- sx

Decrypted msg: instrumentsx
```