NKAFU VANIC ASONG

FE21A274

CEF350 - Security and Cryptosystems

Lab Exercises

Exercise 1 – Implementation of the Columnar Transposition cipher

CODE

```
//Encryption Function:
#include <stdio.h>
#include <string.h>

void encrypt(char *plaintext, int n, int *f, char *ciphertext) {
   int i, j, k, len;
   char columns[n][strlen(plaintext)/n+1];
   len = strlen(plaintext);
   if (len % n != 0) {
      len += n - len % n;
      plaintext = realloc(plaintext, len+1);
      memset(plaintext+strlen(plaintext), ' ', len-strlen(plaintext));
   }
   k = 0;
   for (i = 0; i < n; i++) {
      for (j = 0; j < len/n; j++) {
         columns[f[i]][j] = plaintext[k++];
      }
```

```
    }
    k = 0;
    for (i = 0; i < len/n; i++) {
        for (j = 0; j < n; j++) {
            ciphertext[k++] = columns[j][i];
        }
    }
    ciphertext[k] = '0';
}
//Decryption Function:
#include <stdio.h>
#include <string.h>

void decrypt(char *ciphertext, int n, int *f, char *plaintext) {
    int i, j, k, len;
    char rows[n][strlen(ciphertext)/n+1];
    len = strlen(ciphertext);
    k = 0;
    for (i = 0; i < len/n; i++) {
        for (j = 0; j < n; j++) {
            rows[j][i] = ciphertext[k++];
        }
    }
    char temp[n][strlen(ciphertext)/n+1];
    for (i = 0; i < n; i++) {
```

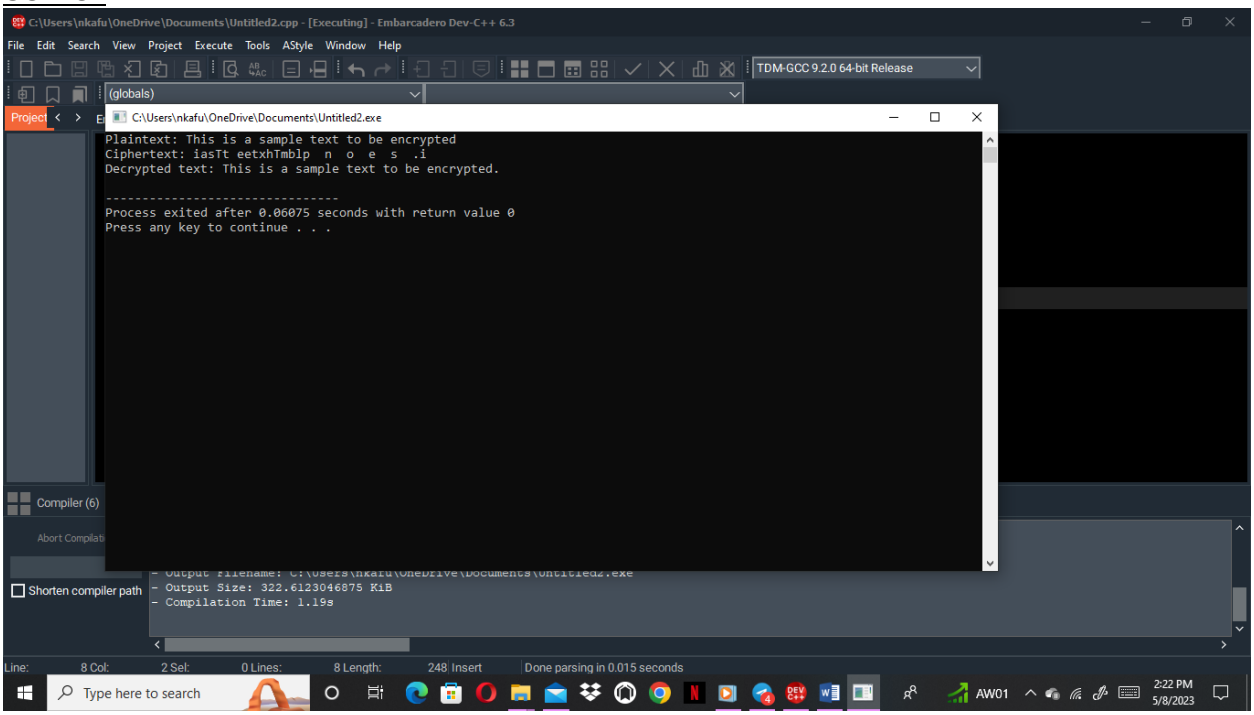
```
      memcpy(temp[i], rows[f[i]], strlen(ciphertext)/n+1);
    }
    memcpy(rows, temp, sizeof(temp));
    k = 0;
    for (i = 0; i < n; i++) {
      for (j = 0; j < len/n; j++) {
        plaintext[k++] = rows[i][j];
      }
    }
    plaintext[k] = '0';
    while (plaintext[k-1] == ' ') {
      plaintext[--k] = '0';
    }
}
return 0;
```

2) Testing with sample text and key

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void encrypt(char *plaintext, int n, int *f, char *ciphertext);
void decrypt(char *ciphertext, int n, int *f, char *plaintext);

int main() {
   char plaintext[] = "This is a sample text to be encrypted.";
   int n = 5;
   int f[] = {2, 4, 0, 3, 1};
   char ciphertext[strlen(plaintext)+1];
   char decryptedtext[strlen(plaintext)+1];
   encrypt(plaintext, n, f, ciphertext);
   printf("Plaintext: %sn", plaintext);
   printf("Ciphertext: %sn", ciphertext);
```

```
    decrypt(ciphertext, n, f, decryptedtext);
    printf("Decrypted text: %sn", decryptedtext);
    return 0;
}
```

OUTPUT



Exercise 2 - Implementation of the Vigenere cipher with key K

1.) CODE

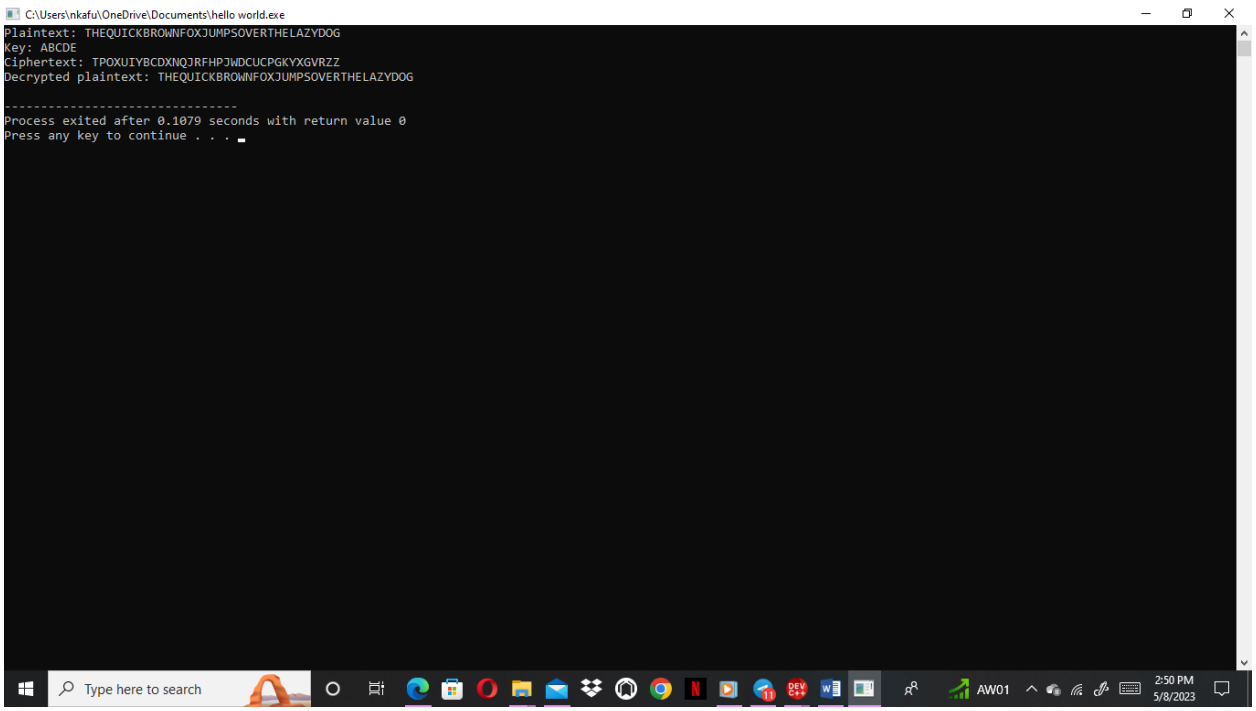
```
#include <stdio.h>
#include <string.h>

void vigenere_encrypt(char *plaintext, char *key, char *ciphertext) {
    int keylen = strlen(key);
    int ptlen = strlen(plaintext);
    int i, j;
    for (i = 0, j = 0; i < ptlen; i++, j = (j + 1) % keylen) {
        int k = key[j] - 'A';
        int p = plaintext[i] - 'A';
        int c = (p + k) % 26;
        ciphertext[i] = c + 'A';
    }
    ciphertext[i] = '\0';
}

void vigenere_decrypt(char *ciphertext, char *key, char *plaintext) {
```

```
    int keylen = strlen(key);
    int ctlen = strlen(ciphertext);
    int i, j;
    for (i = 0, j = 0; i < ctlen; i++, j = (j + 1) % keylen) {
        int k = key[j] - 'A';
        int c = ciphertext[i] - 'A';
        int p = (c - k + 26) % 26;
        plaintext[i] = p + 'A';
    }
    plaintext[i] = '\0';
}
```

2)

```
int main() {
    char plaintext[] = "thequickbrownfoxjumpsoverthelazydog";
    char key[] = "abcde";
    char* ciphertext = vigenere_encrypt(plaintext, key);
    char* decryptedtext = vigenere_decrypt(ciphertext, key);
    printf("Plaintext: %s\n", plaintext);
    printf("Key: %s\n", key);
    printf("Ciphertext: %s\n", ciphertext);
    printf("Decryptedtext: %s\n", decryptedtext);
    free(ciphertext);
    free(decryptedtext);
    return 0;
}
```

OUTPUT


C:\Users\nkafu\OneDrive\Documents\hello world.exe
Plaintext: THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG
Key: ABCDE
Ciphertext: TPOXUIYBCDXNQJRFHPJWDCUCPGKYXGVRZZ
Decrypted plaintext: THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG

Process exited after 0.1079 seconds with return value 0
Press any key to continue . . .
Type here to search
AW01
2:50 PM
5/8/2023