Name: KAMCHE YANN ARNAUD

Matricule: FE21A208

Department: Computer Engineering

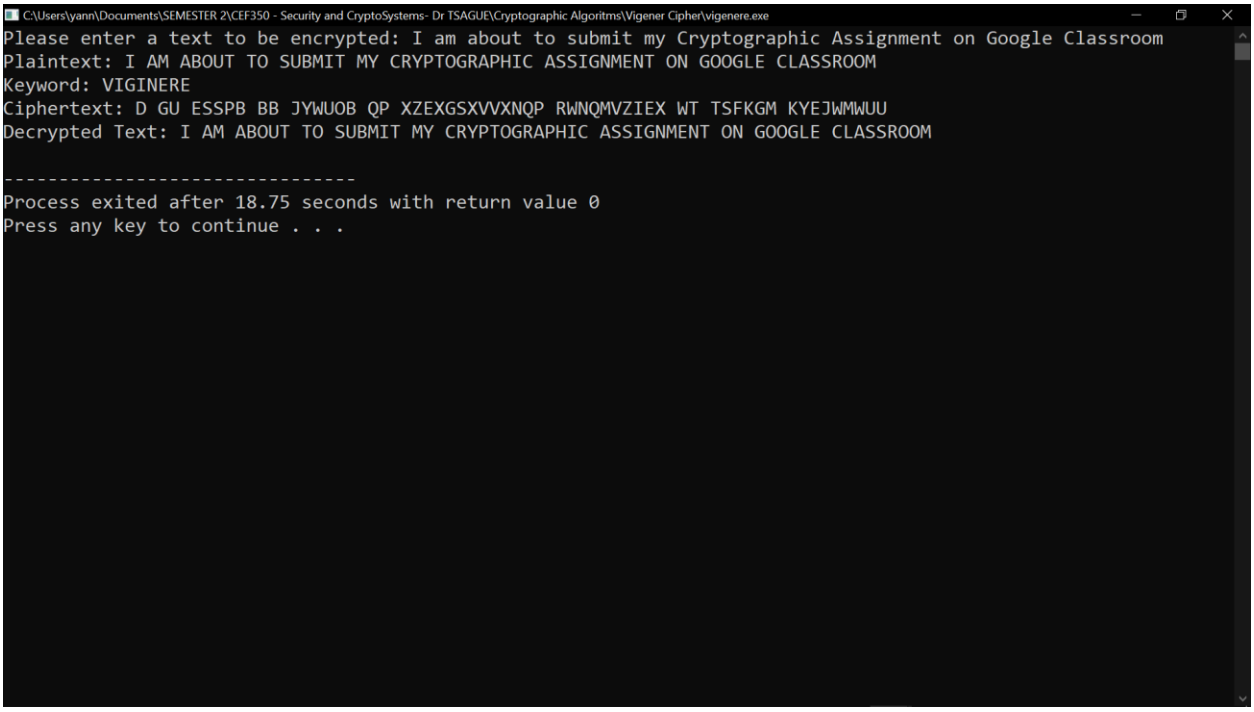
Level: 300

CEF350 - Security and Cryptosystems
Lab Exercises

Exercise 1 – SCREENSHOT AND CODE

Key: "VIGINERE"

Result:



Code:

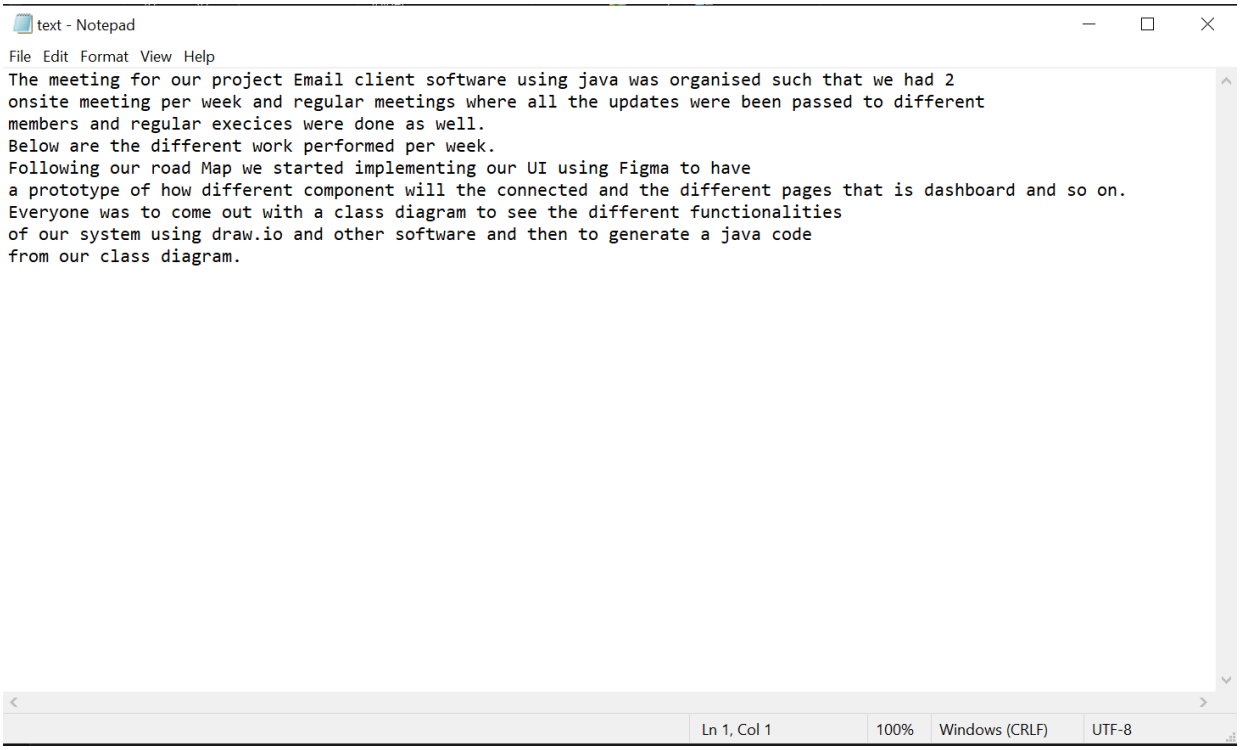
```
/*
Name: KAMCHE YANN ARANAUD
Matricule: FE21A208
*/
#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>
int main()
{
   char inputString[100];
   char keyString [] = "VIGINERE";
        int i, j;
   int keyLength = strlen(keyString);
   printf("Please enter a message: ");
   gets(inputString);
   int stringLength = strlen(inputString);
   char newKeyValue[stringLength], encryptedString[stringLength],
decryptedString[stringLength];


   //Setting inputString to uppercase
   for(i=0; inputString[i]!='\0'; i++)
   {
        if(inputString[i]>='a' && inputString[i]<='z')
        {
                inputString[i] = inputString[i] - 32;
                }
        }
   for(i=0, j=0; i<stringLength; i++, j++)
   {
        if(j == keyLength)
                j=0;
                newKeyValue[i] = keyString[j];
        }
        newKeyValue[i] = '\0';
        for(i=0; i< stringLength; i++)
                if(inputString[i] == 32) //Handling blank space between words
```

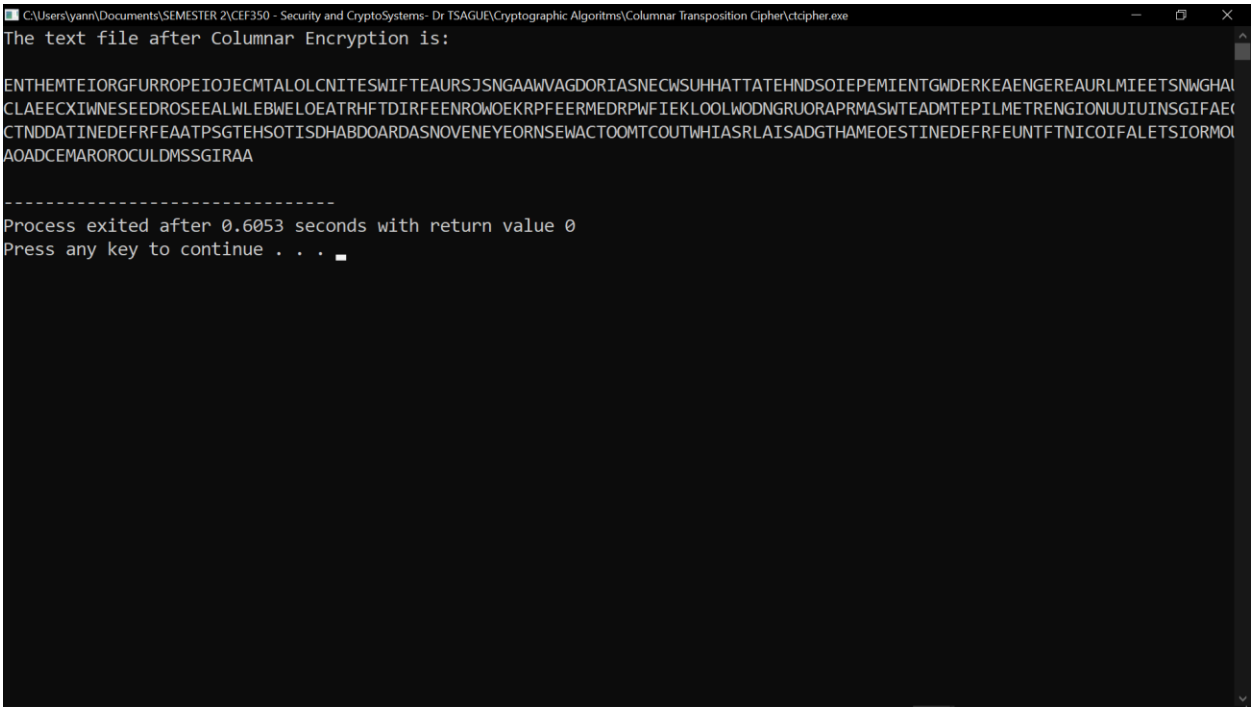
```
            {
                encryptedString[i] = 32; // assigning blank space to encrypted
string
                    }
                    else
                    {


            encryptedString[i] = ((inputString[i] + newKeyValue[i]) % 26) + 65;
}
        encryptedString[i] = '\0'; //Letting the code know that it reached the end of
the string
        for(i=0; i<stringLength; i++)
            if(encryptedString[i] == 32) //Handling blank space between word
            {
                decryptedString[i] = 32; // assigning blank space to decrypted
string
            }
            else
            }
            decryptedString[i] = (((encryptedString[i] - newKeyValue[i]) + 26) %
26) + 65;
}
        decryptedString[i] = '\0';
    //Output results
    printf("Plaintext: %s\n", inputString);
    printf("Keyword: %s\n", keyString);
    printf("Ciphertext: %s\n", encryptedString);
    printf("Decrypted Text: %s\n", decryptedString);
    return 0;
}
```

Exercise 2 = SCREENSHOT AND CODE

Key: "ENCRYPTED"



Result



Code:

```
/*
Name: KAMCHE YANN ARNAUD
Matricule: FE21A208
*/
#include <stdio.h>
#include <stdlib.h>
char key_word[] = {"ENCRYPTED"};
int table_size;
struct table {
    char letter;
};
struct table *array;
int string_length(char *str) {
    int n = 0;
    while (str[n] != '\0') {
        n++;
    }
    return n;
}
char convert_uppercase(char c) {
    char new;
       if (96 < c && c < 123) {
        new = c - 32;
    } else {
        new = c;
    }
    return new;
}
```

```
void init_table() {
    if (table_size % string_length(key_word) != 0) {
        table_size += string_length(key_word) - (table_size %
string_length(key_word));
    }
    array = malloc(table_size * sizeof(char));
    for (int i = 0; i < table_size; i++) {
        // replaces remaining space with character X
        array[i].letter = 'X';
    }
}
void reorder_array(int i, int j) {
    // split into rows of length = keyword
    // do 8 - n/m for position
    // get first row
  int row_count = 0;
    int n = 0;
    int m = 0;
    while (n < table_size){
        if (n - string_length(key_word) * row_count == i) {
            while (m < table_size) {
                if (m - string_length(key_word) * row_count == j) {
                    char temp = array[n].letter;
                    array[n].letter = array[m].letter;
                    array[m].letter = temp;
                    row_count++;
                    m++;
                    n++;
                    break;
                }
```

```
            m++;
          }
        }
        n++;
    }
}
void swap(int i, int j) {
    char temp = key_word[i];
    key_word[i] = key_word[j];
    key_word[j] = temp;
    reorder_array(i, j);
}
int partition(int low, int high) {
    int i = low - 1;
    char pivot = key_word[high];
    for (int j = low; j < high - 1; j++) {
        if (key_word[j] < pivot) {
            i++;
            swap(i, j);
        }
    }
    swap(i + 1, high);

    return i + 1;
}
void sort_chars(int low, int high) {
    if (low < high) {
        int index = partition(low, high);
        sort_chars(low, index - 1);
        sort_chars(index + 1, high);
```

```
    }
}



//
void order_key() {
    sort_chars(0, string_length(key_word) - 1);
}
void read_words_from_file() {
    char *file = "./text.txt";
    FILE *fp = fopen(file, "r");
    // checks if file exists
    if (!fp) {
        printf("\nCan't open file\n");
        return;
    }
    // reads contents of file until end of file
    int i = 0;
    do {
        char c = fgetc(fp);
        if (feof(fp)) {
            break;
        } else if ((96 < c && c < 123) || (64 < c && c < 91)) {
            c = convert_uppercase(c);
            array[i].letter = c;
            i++;
        }
    } while (1);
    fclose(fp);
}
```

```
void get_file_length() {
    char *fileline = "./text.txt";
    FILE *fp = fopen(fileline, "r");
    if (!fp) {
        printf("\nCan't open file\n");
        return;
    }
    do {
        char c = fgetc(fp);

        if (feof(fp)) {
            break;
        } else if ((96 < c && c < 123) || (64 < c && c < 91)) {
            table_size++;
        }
    } while (1);

    fclose(fp);
}
void printList() {
    printf("\n");
    for (int i = 0; i < table_size; i++) {
        printf("%c", array[i].letter);
    }
    printf("\n");
}
int main() {
    get_file_length();
    init_table();
    read_words_from_file();
```

```
    order_key();
    printList();
    return 0;
}
```