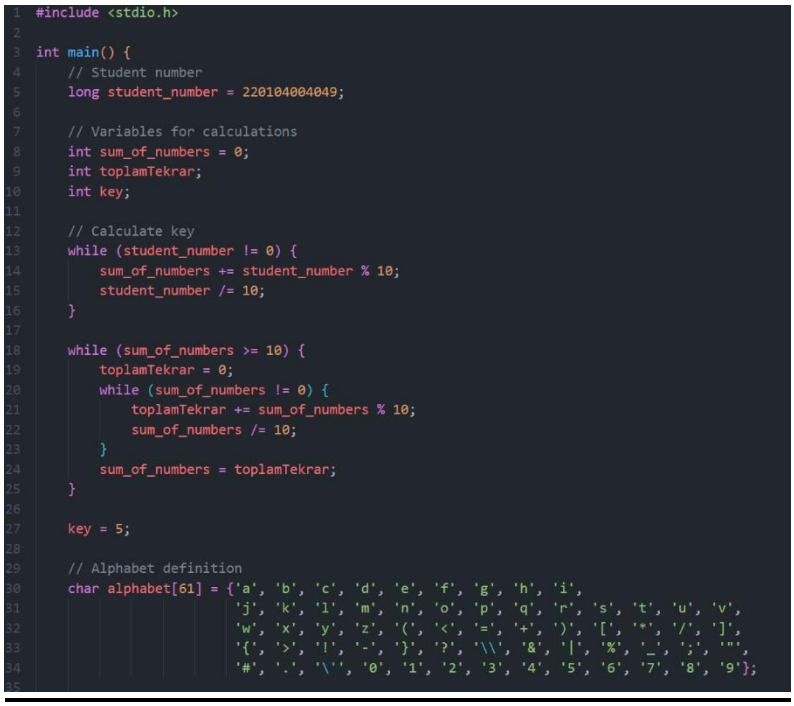
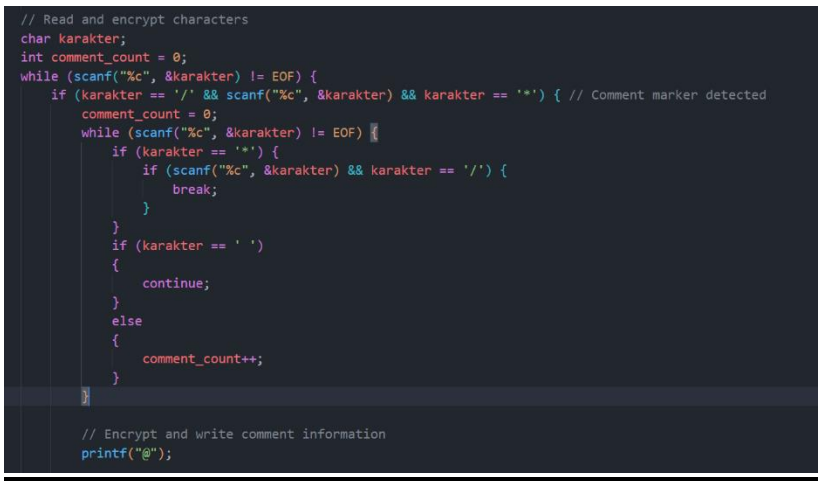
Sample code lines from the homework :

Encrypted :

-The provided C program encrypts text input using a customized encryption method based on the sum of digits of a student number as the encryption key. It reads characters from the input stream, and when it encounters a comment marker '/*', it calculates the length of the comment and encrypts it into two characters. For other characters, it performs a case-insensitive encryption by shifting their positions in a defined alphabet array based on the encryption key. The encrypted output is printed to the console. Additionally, it handles special cases such as uppercase characters and non-alphabetical characters. The program demonstrates basic file I/O, character manipulation, and modular arithmetic operations within a C environment.



```c
#include <stdio.h>

int main() {
    // Student number
    long student_number = 220104004049;

    // Variables for calculations
    int sum_of_numbers = 0;
    int toplamTekrar;
    int key;

    // Calculate key
    while (student_number != 0) {
        sum_of_numbers += student_number % 10;
        student_number /= 10;
    }

    while (sum_of_numbers >= 10) {
        toplamTekrar = 0;
        while (sum_of_numbers != 0) {
            toplamTekrar += sum_of_numbers % 10;
            sum_of_numbers /= 10;
        }
        sum_of_numbers = toplamTekrar;
    }

    key = 5;

    // Alphabet definition
    char alphabet[61] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
                         'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
                         'w', 'x', 'y', 'z', '(', '<', '=', '+', ')', '[', '*', '/', ']',
                         '{', '>', '!', '-', '}', '?', '\\', '&', '|', '%', '_', ';', '"',
                         '#', '.', '\'', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
```



```c
// Read and encrypt characters
char karakter;
int comment_count = 0;
while (scanf("%c", &karakter) != EOF) {
    if (karakter == '/' && scanf("%c", &karakter) && karakter == '*') { // Comment marker detected
        comment_count = 0;
        while (scanf("%c", &karakter) != EOF) {
            if (karakter == '*') {
                if (scanf("%c", &karakter) && karakter == '/') {
                    break;
                }
            }
            if (karakter == ' ')
            {
                continue;
            }
            else
            {
                comment_count++;
            }
        }

        // Encrypt and write comment information
        printf("@");
```

```
            // Onlar basamağını alıp şifreleme
            char onlar = (comment_count / 10) + '0';
            int tmp_index = -1;
            for (int i = 0; i < 61; i++) {
                if (onlar == alphabet[i]) {
                    tmp_index = i;
                    break;
                }
            }

            // Encrypt and write character if found
            if (tmp_index != -1) {
                int new_index = (tmp_index + key) % 61;
                printf("%c", alphabet[new_index]);
            } else {
                // Write non-alphabetical characters directly
                printf("%c", onlar);
            }
            // Birler basamağını alıp şifreleme
            char birler = (comment_count % 10) + '0';
            int tmp_index2 = -1;
            for (int i = 0; i < 61; i++) {
                if (onlar == alphabet[i]) {
                    tmp_index2 = i;
                    break;
                }
            }
```

```
            // Encrypt and write character if found
            if (tmp_index2 != -1) {
                int new_index = (tmp_index2 + key) % 61;
                printf("%c", alphabet[new_index]);
            } else {
                // Write non-alphabetical characters directly
                printf("%c", birler);
            }

        }
        else {
            int index = -1;

            // Convert uppercase to lowercase for case-insensitive encryption
            if (karakter >= 'A' && karakter <= 'Z') {
                karakter = karakter + 32;
            }

            // Find character index in the alphabet
            for (int i = 0; i < 61; i++) {
                if (karakter == alphabet[i]) {
                    index = i;
                    break;
                }
            }
```

```
            // Birler basamağını alıp şifreleme
            char birler = (comment_count % 10) + '0';
            int tmp_index2 = -1;
            for (int i = 0; i < 61; i++) {
                if (onlar == alphabet[i]) {
                    tmp_index2 = i;
                    break;
                }
            }

            // Encrypt and write character if found
            if (tmp_index2 != -1) {
                int new_index = (tmp_index2 + key) % 61;
                printf("%c", alphabet[new_index]);
            } else {
                // Write non-alphabetical characters directly
                printf("%c", birler);
            }

        }
        else {
            int index = -1;

            // Convert uppercase to lowercase for case-insensitive encryption
            if (karakter >= 'A' && karakter <= 'Z') {
                karakter = karakter + 32;
            }

            // Find character index in the alphabet
            for (int i = 0; i < 61; i++) {
                if (karakter == alphabet[i]) {
                    index = i;
                    break;
                }
            }

            // Encrypt and write character if found
            if (index != -1) {
                int new_index = (index + key) % 61;
                printf("%c", alphabet[new_index]);
            } else {
                // Write non-alphabetical characters directly
                printf("%c", karakter);
            }
        }
    }
    return 0;
```

Decrypted :

The provided C program implements a decryption mechanism based on a custom encryption scheme using a student number as the key. The program first calculates the key by summing the digits of the student number and reducing it to a single-digit value. It then utilizes this key to decrypt the input text.

The decryption process involves identifying special markers denoting encrypted comments and decrypting the characters accordingly. When the program encounters the marker '@', it recognizes the beginning of an encrypted comment. The encrypted comment information is read, decrypted using the key, and printed out. Additionally, the program handles uppercase letters by converting them to lowercase before decryption. Non-alphabetical characters are directly printed without decryption.

Overall, the program demonstrates basic file I/O, character manipulation, and decryption techniques within a C environment.

```c
#include <stdio.h>

int main() {
    // Student number
    long student_number = 220104004049;

    // Variables for calculations
    int sum_of_numbers = 0;
    int toplamTekrar;
    int key;

    // Calculate key
    while (student_number != 0) {
        sum_of_numbers += student_number % 10;
        student_number /= 10;
    }

    while (sum_of_numbers >= 10) {
        toplamTekrar = 0;
        while (sum_of_numbers != 0) {
            toplamTekrar += sum_of_numbers % 10;
            sum_of_numbers /= 10;
        }
        sum_of_numbers = toplamTekrar;
    }

    key = 5;

    // Alphabet definition
    char alphabet[61] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
                         'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
                         'w', 'x', 'y', 'z', '(', '<', '=', '+', ')', '[', '*', '/', ']',
                         '{', '>', '!', '-', '}', '?', '\\', '&', '|', '%', '_', ';', '"',
                         '#', '.', '\'', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
```

```c
 char karakter;
 int comment_count = 0;
 while (scanf("%c", &karakter) != EOF) {
     if (karakter == '@') { // Comment marker detected
         // Read encrypted comment information
         printf("/*");
         char onlar, birler;
         scanf("%c%c", &onlar, &birler);

         // Decrypt onlar basamağı
         int tmp_index = -1;
         for (int i = 0; i < 61; i++) {
             if (onlar == alphabet[i]) {
                 tmp_index = i;
                 break;
             }
         }
         printf("There is a ");
         // Decrypt and write character if found
         if (tmp_index != -1) {
             int new_index = (tmp_index - key + 61) % 61;
             printf("%c", alphabet[new_index]);
         } else {
             // Write non-alphabetical characters directly
             printf("%c", onlar);
         }

         // Decrypt birler basamağı
         int tmp_index2 = -1;
         for (int i = 0; i < 61; i++) {
             if (birler == alphabet[i]) {
                 tmp_index2 = i;
                 break;
             }
         }

         // Decrypt and write character if found
         if (tmp_index2 != -1) {
             int new_index = (tmp_index2 - key + 61) % 61;
             printf("%c", alphabet[new_index]);
         } else {
             // Write non-alphabetical characters directly
             printf("%c", birler);
         }
         printf(" characters as comment.");
         printf("*/");
```

```
        }
        else {
            int index = -1;

            // Convert uppercase to lowercase for case-insensitive decryption
            if (karakter >= 'A' && karakter <= 'Z') {
                karakter = karakter + 32;
            }

            // Find character index in the alphabet
            for (int i = 0; i < 61; i++) {
                if (karakter == alphabet[i]) {
                    index = i;
                    break;
                }
            }

            // Decrypt and write character if found
            if (index != -1) {
                int new_index = (index - key + 61) % 61;
                printf("%c", alphabet[new_index]);
            } else {
                // Write non-alphabetical characters directly
                printf("%c", karakter);
            }
        }
    }
    return 0;
}
```

The example outputs of the encrypted and decrypted code :

```
#include <stdio.h>

int main()
{
    /*multiply 2 numbers*/
    int x = 3;
    int y = 4;
    int result = x * y;
    return 0;
}
```

```
ynalbant@DESKTOP-UV42VUV:~/assignment1$ ./a.out < another_input.txt > another_output.txt
```

```
2nshqzij *xyint3m\

nsy rfns[{
?
    @66
    nsy = / 80
    nsy + / 90
    nsy wjxzqy / = ! +0
    wjyzws 50
%
```

```
ynalbant@DESKTOP-UV42VUV:~/assignment1$ ./a.out < another_output.txt > another_input.txt
```

```c
#include <stdio.h>

int main()
{
    /*There is a 11 characters as comment.*/
    int x = 3;
    int y = 4;
    int result = x * y;
    return 0;
}
```

```c
#include <stdio.h>
int main() {/*This is the main function.*/
int number_one = 3;
int number_two = 4;
int add = number_one + number_two;
return 0;
}
```

```
ynalbant@DESKTOP-UV42VUV:~/assignment1$ ./a.out < input.txt > output.txt
```

```
2nshqzij *xyint3m\
nsy rfns[{ ?@77
nsy szrgjw'tsj / 80
nsy szrgjw'y<t / 90
nsy fii / szrgjw'tsj ] szrgjw'y<t0
wjyzws 50
%
```

```
ynalbant@DESKTOP-UV42VUV:~/assignment1$ ./a.out < output.txt > input.txt
```

```c
#include <stdio.h>
int main() {/*There is a 22 characters as comment.*/
int number_one = 3;
int number_two = 4;
int add = number_one + number_two;
return 0;
}
```

Youtube Link :

https://youtu.be/CbsUTWuWm2E