

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.

```
1 #include <stdio.h>
2
3 int main() {
4     // Student number
5     long student_number = 220104004049;
6
7     // Variables for calculations
8     int sum_of_numbers = 0;
9     int toplamTekrar;
10    int key;
11
12    // Calculate key
13    while (student_number != 0) {
14        sum_of_numbers += student_number % 10;
15        student_number /= 10;
16    }
17
18    while (sum_of_numbers >= 10) {
19        toplamTekrar = 0;
20        while (sum_of_numbers != 0) {
21            toplamTekrar += sum_of_numbers % 10;
22            sum_of_numbers /= 10;
23        }
24        sum_of_numbers = toplamTekrar;
25    }
26
27    key = 5;
28
29    // Alphabet definition
30    char alphabet[61] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
31                        'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
32                        'w', 'x', 'y', 'z', '(', '<', '=', '+', ')', '[', '*', '/', ']',
33                        '{', '>', '!', '-', '}', '?', '\\', '&', '|', '%', '_', ';', '"',
34                        '#', '.', '\\', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
35}
```

```
// 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("@");
    }
}
```

```

61 // Onlar basamağını alıp şifreleme
62 char onlar = (comment_count / 10) + '0';
63 int tmp_index = -1;
64 for (int i = 0; i < 61; i++) {
65     if (onlar == alphabet[i]) {
66         tmp_index = i;
67         break;
68     }
69 }
70
71 // Encrypt and write character if found
72 if (tmp_index != -1) {
73     int new_index = (tmp_index + key) % 61;
74     printf("%c", alphabet[new_index]);
75 } else {
76     // Write non-alphabetical characters directly
77     printf("%c", onlar);
78 }
79 // Birler basamağını alıp şifreleme
80 char birler = (comment_count % 10) + '0';
81 int tmp_index2 = -1;
82 for (int i = 0; i < 61; i++) {
83     if (onlar == alphabet[i]) {
84         tmp_index2 = i;
85         break;
86     }
87 }

```

```

89 // Encrypt and write character if found
90 if (tmp_index2 != -1) {
91     int new_index = (tmp_index2 + key) % 61;
92     printf("%c", alphabet[new_index]);
93 } else {
94     // Write non-alphabetical characters directly
95     printf("%c", birler);
96 }
97
98 }
99 else {
100     int index = -1;
101
102     // Convert uppercase to lowercase for case-insensitive encryption
103     if (karakter >= 'A' && karakter <= 'Z') {
104         karakter = karakter + 32;
105     }
106
107     // Find character index in the alphabet
108     for (int i = 0; i < 61; i++) {
109         if (karakter == alphabet[i]) {
110             index = i;
111             break;
112         }
113     }
114 }

```

```

79 // Birler basamağını alıp şifreleme
80 char birler = (comment_count % 10) + '0';
81 int tmp_index2 = -1;
82 for (int i = 0; i < 61; i++) {
83     if (onlar == alphabet[i]) {
84         tmp_index2 = i;
85         break;
86     }
87 }
88
89 // Encrypt and write character if found
90 if (tmp_index2 != -1) {
91     int new_index = (tmp_index2 + key) % 61;
92     printf("%c", alphabet[new_index]);
93 } else {
94     // Write non-alphabetical characters directly
95     printf("%c", birler);
96 }
97
98 }
99 else {
100     int index = -1;
101
102     // Convert uppercase to lowercase for case-insensitive encryption
103     if (karakter >= 'A' && karakter <= 'Z') {
104         karakter = karakter + 32;
105     }
106
107     // Find character index in the alphabet
108     for (int i = 0; i < 61; i++) {
109         if (karakter == alphabet[i]) {
110             index = i;
111             break;
112         }
113     }
114
115     // Encrypt and write character if found
116     if (index != -1) {
117         int new_index = (index + key) % 61;
118         printf("%c", alphabet[new_index]);
119     } else {
120         // Write non-alphabetical characters directly
121         printf("%c", karakter);
122     }
123 }
124 }
125 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.

```

#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'};

```

```

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 :

```

1  #include <stdio.h>
2
3  int main()
4  {
5      /*multiply 2 numbers*/
6      int x = 3;
7      int y = 4;
8      int result = x * y;
9      return 0;
10 }

```

```

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

```

```

1  2nshqzij *xyint3m\
2
3  nsy rfns[{
4  ?
5  |
6  | @66
7  | nsy = / 80
8  | nsy + / 90
9  | nsy wjxzqy / = ! +0
10 | wjyzws 50
    %|

```

```

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

```

```

1  #include <stdio.h>
2
3  int main()
4  {
5      /*There is a 11 characters as comment.*/
6      int x = 3;
7      int y = 4;
8      int result = x * y;
9      return 0;
10 }

```

```

#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

```

```

1  2nshqzij *xyint3m\
2  nsy rfns[{ ?@77
3  nsy szrgjw'tsj / 80
4  nsy szrgjw'y<t / 90
5  nsy fii / szrgjw'tsj ] szrgjw'y<t0
6  wjyzws 50
7  %

```

```

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

```

```

1  #include <stdio.h>
2  int main() { /*There is a 22 characters as comment.*/
3  int number_one = 3;
4  int number_two = 4;
5  int add = number_one + number_two;
6  return 0;
7  }

```

Youtube Link :

<https://youtu.be/CbsUTWuWm2E>