CSE 102 HW1 REPORT

Mehmet Akif Pekşen

230104004013

Part 1

```c
#include <stdio.h>

int main()
{

    /* Define the alphabet array with characters and special symbols */
    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',

    /* Declare variables */
    int i, alphabet_counter = 0, comment_start_counter = 0, comment_finish_counter = 0, comment_counter = 0, crypted_number;
    char character;
    int key = 9; /* Define the encryption key. The digits of my school number (230104004013) are 18 in total and the digits of 18 are 9 in total.*/
```

In this section, I define the alphabet array and other variables that I will use in my code. The key variable here is equal to the sum of the digits of 18, which is the sum of the digits of my school number 230104004013, which is 9.

```c
/* Loop until end of file */
while (scanf("%c", &character) != EOF)
{
```

In this section, I define the while loop that will read the characters until the end of the file and store the read value as char.

```c
/* Check if a comment has started */
if (comment_start_counter == 0)
{

    /* If character is '/', it might indicate the start of a comment */
    if (character == '/')
    {
        comment_start_counter = 1;
        continue;
    }

    /* Reset alphabet counter */
    alphabet_counter = 0;

    /* Iterate through the alphabet array to find matches */
    for (i = 0; i < 61; i++)
    {

        /* If character is found in the alphabet array, encrypt and print */
        if (character == alphabet[i])
        {
            alphabet_counter++;
            crypted_number = ((i + key) % 61);
            printf("%c", alphabet[crypted_number]);
        }
    }

    /* If character not found in the alphabet array, print it as it is */
    if (alphabet_counter == 0)
    {
        printf("%c", character);
    }
}
```

In this section, it is checked to see if there is a possibility of starting a comment line while reading the code. If not, the encryption process takes place. If the character read is not in the alphabet, it is written as it is. If there is a possibility of starting a comment line, comment_start_counter is set equal to 1 and the cycle is skipped.

```
/* Check the possibility of starting a comment line if the previous character is '/' */
else if (comment_start_counter == 1)
{

    /* If character is '*', it might indicate the start of a multi-line comment */
    if (character == '*')
    {
        comment_start_counter = 2;
        printf("@ "); /* Placeholder for the encrypted comment start */
    }
    else
    {
        /* Print encrypted character for '/' and reset comment counter */
        printf("%c", alphabet[33 + key]);
        comment_start_counter = 0;
        for (i = 0; i < 61; i++)
        {
            if (character == alphabet[i])
            {
                crypted_number = ((i + key) % 61);
                printf("%c", alphabet[crypted_number]);
            }
        }
    }
}
```

In this section, if the immediately previous read character is '/', it is entered, that is, the possibility of starting the comment line is checked. If the read character is '*', it means that the comment line has started. comment_start_counter is equal to 2. If the read value is not '*', the counter is reset and the encrypted version of the '/' expression (the previous read '/' was not encrypted or printed) and the encrypted version of the currently read character is printed.

```
        /* If a comment has started */
        else if (comment_start_counter == 2)
        {

            /* Check for the end of the comment */
            if (character == '*' && comment_finish_counter == 0)
            {
                comment_finish_counter = 1;
            }

            else if (character == '/' && comment_finish_counter == 1)
            {
                comment_finish_counter = 2;
            }

            else if (comment_finish_counter == 2)
            {
                /* Placeholder for the encrypted comment end */
                switch ((comment_counter / 10) % 10)
                {
                case 0:
                    printf("%c", alphabet[(51 + key) % 61]);
                    break;
                case 1:
                    printf("%c", alphabet[(52 + key) % 61]);
                    break;
                case 2:
                    printf("%c", alphabet[(53 + key) % 61]);
                    break;
                case 3:
                    printf("%c", alphabet[(54 + key) % 61]);
                    break;
                case 4:
                    printf("%c", alphabet[(55 + key) % 61]);
                    break;
```

```
                case 5:
                    printf("%c", alphabet[(56 + key) % 61]);
                    break;
                case 6:
                    printf("%c", alphabet[(57 + key) % 61]);
                    break;
                case 7:
                    printf("%c", alphabet[(58 + key) % 61]);
                    break;
                case 8:
                    printf("%c", alphabet[(59 + key) % 61]);
                    break;
                case 9:
                    printf("%c", alphabet[(60 + key) % 61]);
                    break;
                }

                switch ((comment_counter) % 10)
                {
                case 0:
                    printf("%c", alphabet[(51 + key) % 61]);
                    break;
                case 1:
                    printf("%c", alphabet[(52 + key) % 61]);
                    break;
                case 2:
                    printf("%c", alphabet[(53 + key) % 61]);
                    break;
                case 3:
                    printf("%c", alphabet[(54 + key) % 61]);
                    break;
                case 4:
                    printf("%c", alphabet[(55 + key) % 61]);
                    break;
                case 5:
                    printf("%c", alphabet[(56 + key) % 61]);
```

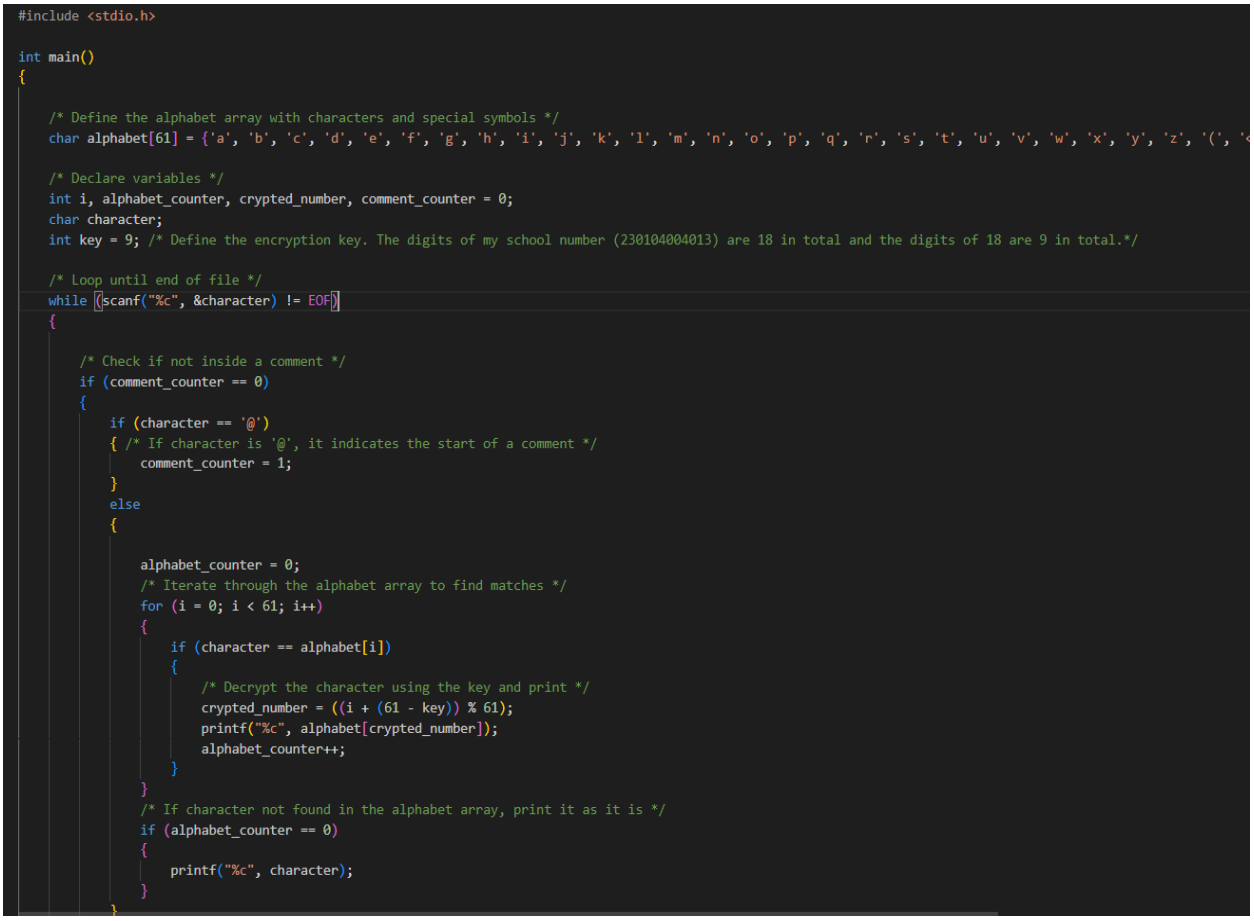
```
                    break;
                case 6:
                    printf("%c", alphabet[(57 + key) % 61]);
                    break;
                case 7:
                    printf("%c", alphabet[(58 + key) % 61]);
                    break;
                case 8:
                    printf("%c", alphabet[(59 + key) % 61]);
                    break;
                case 9:
                    printf("%c", alphabet[(60 + key) % 61]);
                    break;
                }
                printf("\n");
                comment_start_counter = 0;
                comment_finish_counter = 0;
                comment_counter = 0;
            }

            else
            {
                /* Counts the number of characters in the comment line */
                if (character != ' ')
                {
                    comment_counter++;
                }
            }
        }
    }

    return 0;
}
```

In this section, the comment line is read. If the character read in this section is '*', it means there is a possibility that the comment line will end. The comment_finish_counter variable is set to 1. If comment_finish_counter is 1 and the next character is '/', comment_finish_counter is set to 2, which means the comment line has ended. . Except for these cases, if the character read is a character other than a space, the comment_counter variable is incremented by one. These processes continue until the comment line ends. When the comment line ends, that is, when comment_finish_counter becomes 2, the numbers in the tens and ones place of the comment_counter variable are encrypted separately with a switch case.

Part 2



```c
#include <stdio.h>

int main()
{

    /* Define the alphabet array with characters and special symbols */
    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', '(', '<

    /* Declare variables */
    int i, alphabet_counter, crypted_number, comment_counter = 0;
    char character;
    int key = 9; /* Define the encryption key. The digits of my school number (230104004013) are 18 in total and the digits of 18 are 9 in total.*/

    /* Loop until end of file */
    while (scanf("%c", &character) != EOF)
    {

        /* Check if not inside a comment */
        if (comment_counter == 0)
        {
            if (character == '@')
            { /* If character is '@', it indicates the start of a comment */
                comment_counter = 1;
            }
            else
            {

                alphabet_counter = 0;
                /* Iterate through the alphabet array to find matches */
                for (i = 0; i < 61; i++)
                {
                    if (character == alphabet[i])
                    {
                        /* Decrypt the character using the key and print */
                        crypted_number = ((i + (61 - key)) % 61);
                        printf("%c", alphabet[crypted_number]);
                        alphabet_counter++;
                    }
                }
                /* If character not found in the alphabet array, print it as it is */
                if (alphabet_counter == 0)
                {
                    printf("%c", character);
                }
            }
```

In this part, the opposite of the encryption process in the first part takes place. If the character read is not in the alphabet, it is printed as it is. If the read value is the @ sign, the comment line shows that it is encrypted and the comment_counter is set equal to 1.

```
        /* If inside a comment */
        else if (comment_counter == 1)
        {
            alphabet_counter = 0;
            for (i = 0; i < 61; i++)
            {
                if (character == alphabet[i])
                {
                    printf("/* There is ");
                    /* Decrypt the character using the key and print */
                    crypted_number = ((i + (61 - key)) % 61);
                    printf("%c", alphabet[crypted_number]);
                    alphabet_counter++;
                    comment_counter = 2;
                }
            }
        }
        else if (comment_counter == 2)
        {
            alphabet_counter = 0;
            for (i = 0; i < 61; i++)
            {
                if (character == alphabet[i])
                {
                    /* Decrypt the character using the key and print */
                    crypted_number = ((i + (61 - key)) % 61);
                    printf("%c", alphabet[crypted_number]);
                    alphabet_counter++;
                    printf(" characters as comment */");
                    comment_counter = 0;
                }
            }
            /* If character not found in the alphabet array, print it as it is */
            if (alphabet_counter == 0)
            {
                printf("%c", character);
            }
        }
    }

    return 0;
}
```

In the first part, the tens digit of the character number of the encrypted comment line is decrypted, and in the second part, the ones digit is decrypted and added as a comment line as requested in the assignment.

Input and Output

```
ktop/CSE102/HW1$ gcc --ansi 230104004013_part1.c -o 1
ktop/CSE102/HW1$ gcc --ansi 230104004013_part2.c -o 2
ktop/CSE102/HW1$ ./1 <input.txt>encrypted.txt
ktop/CSE102/HW1$ ./2 <encrypted.txt>decrypted.txt
```

input.txt:

```
#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;
```

encrypted.txt:

```
6rwlu+mn ><=mrx7q_

rw= vjrw{}% @ bb
    rw= w+vkn(3xwn ! c4
    rw= w+vkn(3=[x ! d4
    rw= jmm ! w+vkn(3xwn - w+vkn(3=[x4
    (n=+(w 94
#
```

decrypted.txt:

```
#include <stdio.h>

int main(){ /* There is 22 characters as comment */
    int number_one = 3;
    int number_two = 4;
    int add = number_one + number_two;
    return 0;
}
```

```
ktop/CSE102/HW1$ ./1 <input_test.txt>encrypted.txt
ktop/CSE102/HW1$ ./2 <encrypted.txt>decrypted.txt
```

input_test.txt:

```
#include <stdio.h>

int main(){
    int num1 = 11, num2 = 5;
    int num3 = num1 / num2; /*This is result.*/
    int num4 = num2 * num3 / num1 ; /*This is another result.*/
    return 0;
}
```

encrypted.txt:

```
6rwlu+mn ><=mrx7q_

rw= vjrw{}%
    rw= w+va ! aa, w+vb ! e4
    rw= w+vc ! w+va &w+vb4 @ ac
    rw= w+vd ! w+vb \ w+vc &w+va 4 @ b9
    (n=+(w 94
#
```

decrypted.txt:

```
#include <stdio.h>

int main(){
    int num1 = 11, num2 = 5;
    int num3 = num1 /num2; /* There is 13 characters as comment */
    int num4 = num2 * num3 /num1 ; /* There is 20 characters as comment */
    return 0;
}
```

YouTube Link:

https://youtu.be/f3RAnchdkcw