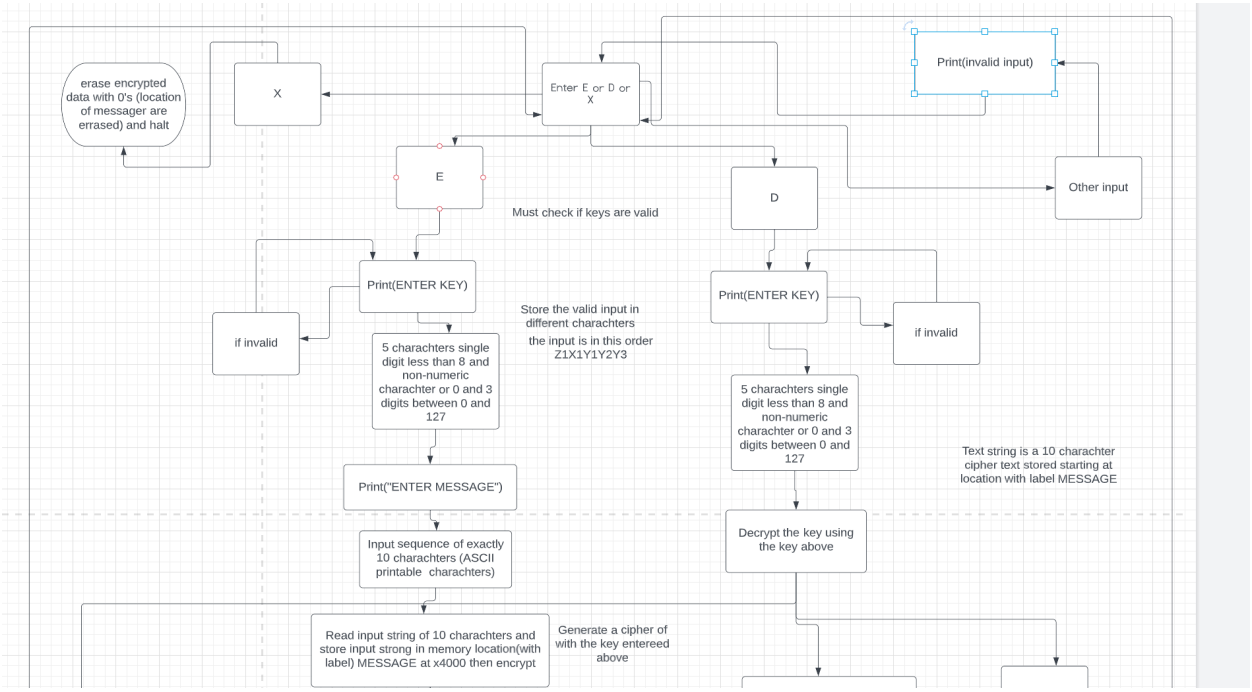
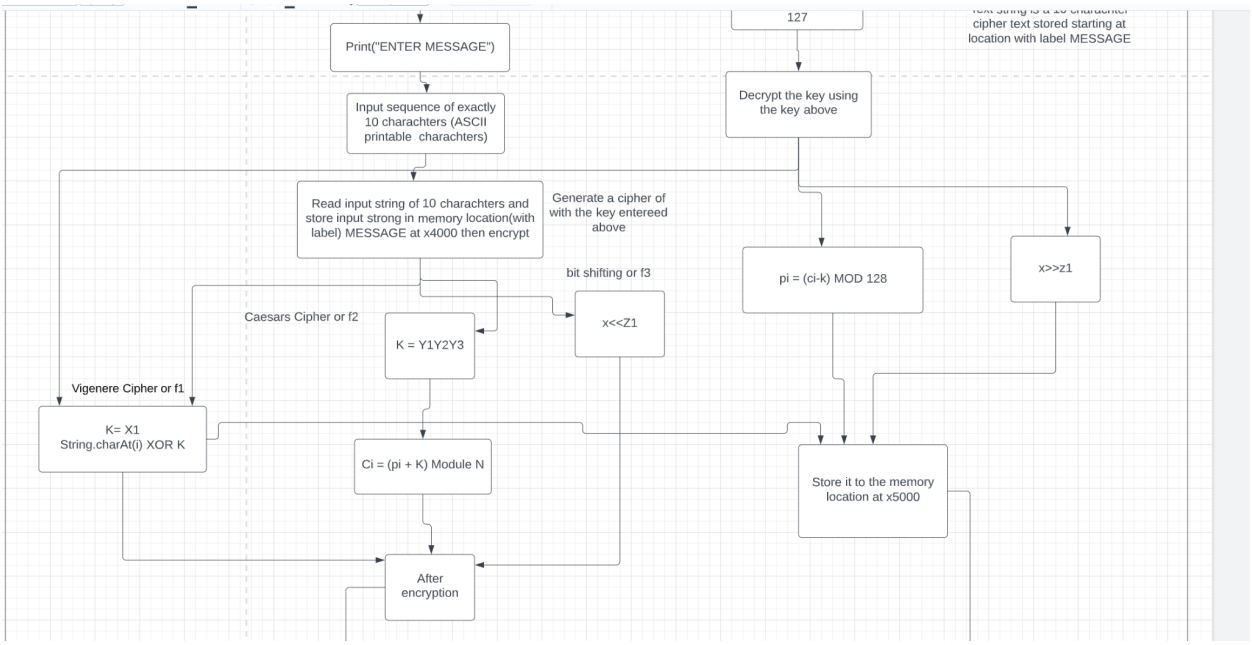
Project 4 report

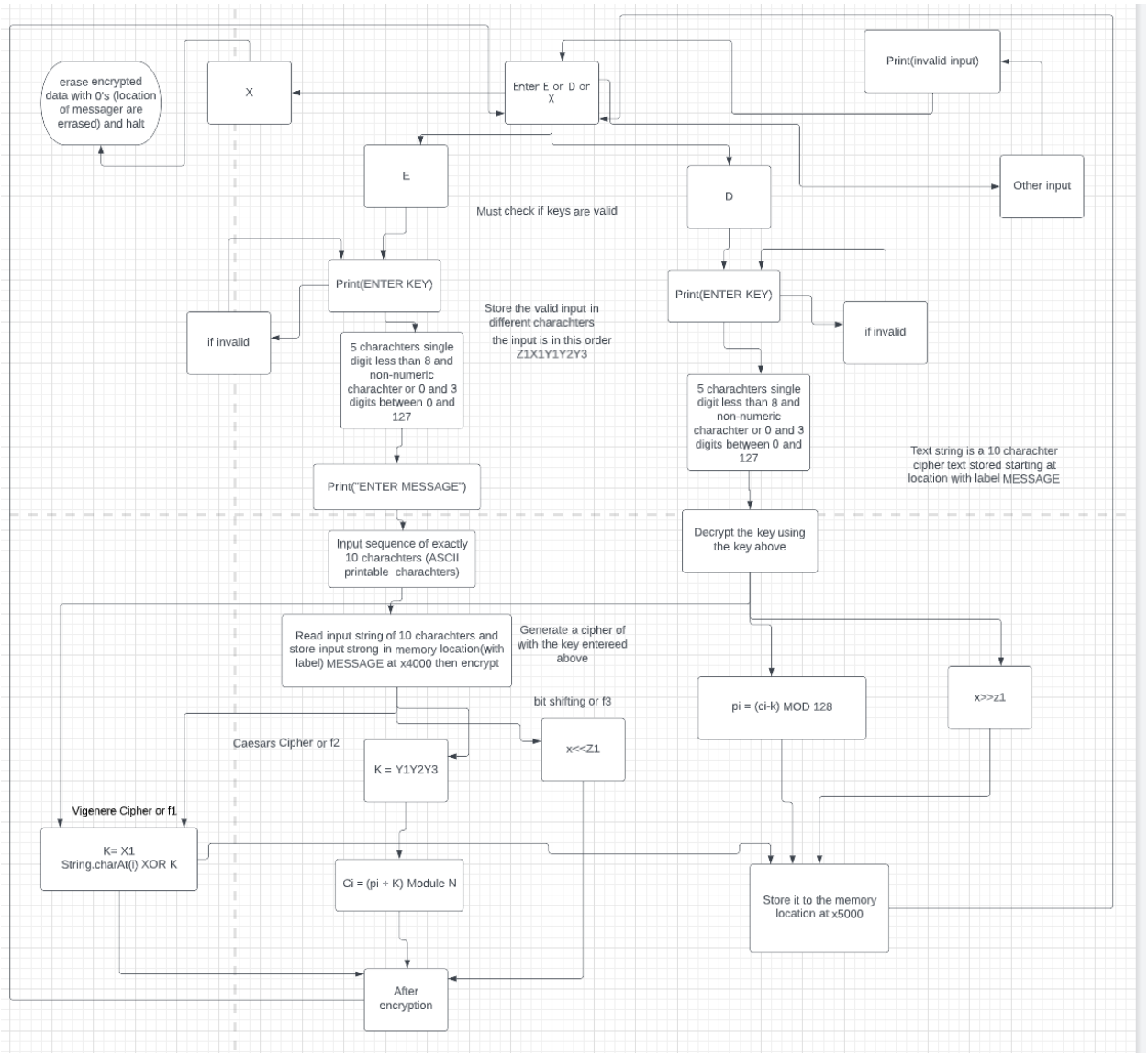
My flowchart it is divided into two parts to make it clear for the reader





(0) The flowchart

The flowchart above is divided into two pictures and the one after this is the whole thing



The link to the flowchart is https://lucid.app/lucidchart/a56d6d3f-20f2-4b54-8bfb-e38552bd9a94/edit?viewport_loc=-1478%2C309%2C4145%2C2128%2C0_0&invitationId=inv_7672a293-b3b9-4d1c-b4d5-a6767c8984b1

(1) The prompt

In this part I first printed out the message that they wanted. After that I also printed(ENTER E OR D OR X) then I took an input And checked if it was D(if it is D go to (#3 decryption)) or E(if it is E go to (#4 encryption) or X(if it is X then I would remove everything in address x4000 to x4009 (where the encrypted message is supposed to be and then halts. I did the input checking by storing the ascii value of E D and X and comparing them with the input if the result is zero then the input is the same as the value compared. Any other input would result to start the program from (ENTER E OR D OR X). What ever is entered is stored in a variable called SaveR0(saveR0 is a variable that I used with BLKW)

(2) The key

In either encrypt or decrypt
I use getc 5 times for each one (Z1 X1 Y1 Y2 Y3)

Z1

i check the first input to check if its more than 7 or less than 0 if either one is true then I add 1 to R6(this register is to repeat either encrypt or decrypt whatever was called then say invalid input)
I then check the second input in this input there are two cases

X1

FirstI check if its less than 9 or equal (case 1) I then go to case1 in the code and then check if its less than 0 or equal to 0 if it is then its valid else add R6 +=1;
Second i check if its more than 0 (case 2) I then go to case2 in the code and then check if its more than 9 if it is its valid else R6+=1;

Y1Y2Y3

For Y1 i checked if it was zero if it is then check if y2 and y3 are any number(if not R6+=1;) however if Y1 was 1 then I checked if y2 and do the same thing as i did with Y1 if it is 1 or 0 then just check if y3 is a number(if not R6+=1;) if it is 2 then i checked if R3 is a number between 0 to 7 (if not R6+=1).
If Y2 is not 0,1, or 2 then R6+=1. If Y1 is not 1 or 0 then R6+=1(which would result in an invalid input)

After checking the key I just do a branch if R6 is positive to beginning of asking the for the key again and doing the whole thing again
If R6 is 0 then I would save the values of each key Z1 would go to SaveZ1 (i used LC3 BLKW) to store the value. And I did it to every key for example, X1 would be stored in SaveX1

(3) Decryption

I first asked for the key(2)

F1

For the decryption of an XOR we can use XOR because in the truth table below we can see that An XOR of an XOR is an XOR
We can use the XOR scheme again and it would work. An example of this is lets assume we want to do #41 XOR #70 this would result in #111. And if we did #41 XOR #111 = #70. Since it returned to the original result we know we can use XOR again for decryption. Also, we can prove this by a table

K below is the result of a xor b

a	b	k
0	0	0
0	1	1
1	0	1
1	1	0

a	K	b
0	0	1
0	1	0
1	1	1
1	0	0

As seen above the xor of an xor is xor. This is how i decrypted the message i just used XOR
I would get the value at x5000 it works because f3 is always first so I can use it like this. Let us assume that a variable l is at address x5000 so we would do **l xor X1**

Then I would store result in address x5000 and I would do that until x5009

F2

First I get the value K which is =y1y2y3. I do this by first adding y3 to one register then I do a while loop and ADD 10 for each y2 Finally if y1 is 1 I add 100 if Y1 is 0 i just do nothing.

pi = (ci-k) MOD 128 (Ci is the values x5000's and k is the key = y1y2y3)
And then i would store the result pi x5000's
I do the mod by doing it like in HW4

F3

This would have to be the first decryption because the number is going to be too big and any other encryption would lose the value of a number. So, we need to implement this first in the decryption method.

Right shift implementation is done by doing a while loop and decreasing the value by 2 until it reaches 0 or less and incrementing another register. Then I would do that Z1 times. The following C code is what i did but i did it in LC3 It is good to visualize it. Sum is the value stored at address x5000 and z1 is input z1.

```
int z1 = 4;
int sum = 480;
int temp = sum;
int solution = 0;
while (z1>0){
   solution = 0;
   sum = temp;
   while (sum > 0){
      solution++;
      temp = solution;
      sum = sum - 2;
   }

   z1--;
}
printf("%d\n", solution);//this would be equal to 480
```

Then I would store solution on address x5000 to x5009

(4) Encryption

I first asked for the key(2) After that

Message

After storing the key then I would print ENTER MESSAGE in the console then I would use getc 10 times and storing the character each time x4000 then x4001 until x4009. And store the values in m0, m1,...., until m9 (m0 through m9 are variables that save the current message character)

F1

In this method I just used what we did in XOR in HW4 and implemented it.
The goal is to do
m(i) XOR Z1 (where m(i) is message at address x4000 to x4009 and z1 is the 2nd key)

I then Store the results in Address x4000 to Address x4009 and store the values in m0 through m9 where I store the message

F2

First I get the value K which is =y1y2y3. I do this by first adding y3 to one register then I do a while loop and ADD 10 for each y2 Finally if y1 is 1 I add 100 if Y1 is 0 i just do nothing.
After this I do the **Ci = (pi+k) %12**8; (where k is the key and pi is the value at address x4000 through address x4009)
I do the mod by doing it like in HW4 but i adjust it so it fits the function above.

I then Store the results in Address x4000 to Address x4009 and store the values in m0 through m9 where I store the message

F3

Left shift implementation is done by basically doing a while loop and keep adding the value of the number at the address and keep adding it the C code below is the same implementation as i did in my LC3 program. Where i is the input Z1 and sum is the value stored at x4000

```
int i = 4;
int sum = 30;
while (i>0){
        sum = sum + sum;
        i--;
}
printf("%d\n", sum);//sum would be equal to 480
```

I then Store the results in Address x4000 to Address x4009 and store the values in m0 through m9 where I store the message

Part(b) underlined questions

Q1
How do you decrypt a message that has been encrypted using this [XOR] scheme?
As Explained in part 3 F1 We can use the XOR scheme again and it would work. An example of this is lets assume we want to do #41 XOR #70 this would result in #111. And if we did #41 XOR #111 = #70. Since it returned to the original result we know we can use XOR again for decryption.

Q2
How do you shift left in LC3? How do you right shift in LC3?

Left shift implementation is done by basically doing a while loop and keep adding the value of the number at the address and keep adding it the C code below is the same implementation as i did in my LC3 program. Where z1 is the input Z1 and sum is the value stored at x4000

```
int z1 = 4;
int sum = 30;
while (z1>0){
        sum = sum + sum;
        z1--;
}
printf("%d\n", sum);//sum would be equal to 480
```

Right shift implementation is done by doing a while loop and decreasing the value by 2 until it reaches 0 or less and increment another register. Then i would do that Z1 times. The following C code is what i did but i did in LC3 It is good to visualize it

```
int z1 = 4;
int sum = 480;
int temp = sum;
int solution = 0;
while (z1>0){
   solution = 0;
   sum = temp;
   while (sum > 0){
      solution++;
      temp = solution;
      sum = sum - 2;
   }

   z1--;
}
printf("%d\n", solution);//this would be equal to30 the opposite of the above
```