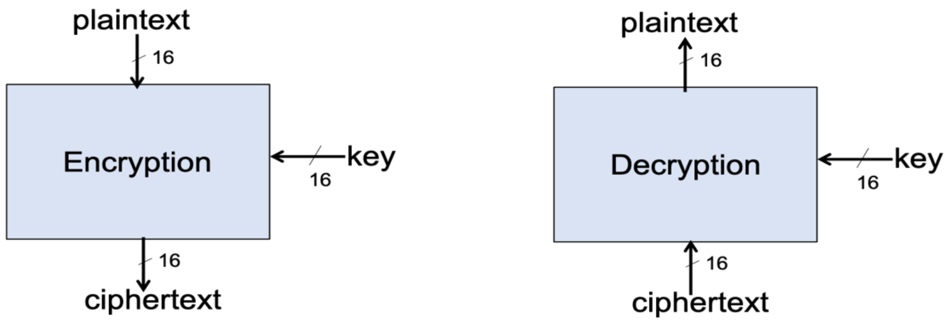
Learning Outcome: Students will gain experience in C programming using strings, loops, pointers, standard input/output, and functions.

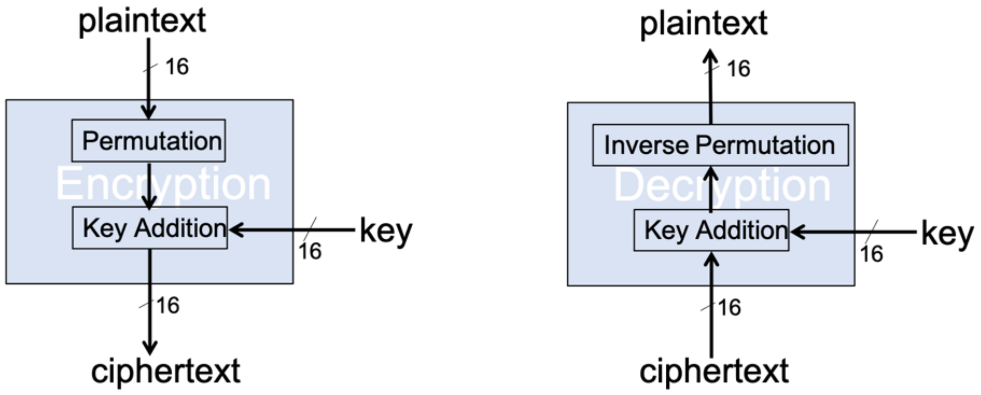
Lab 3: Derp the Cipher

Problem Description

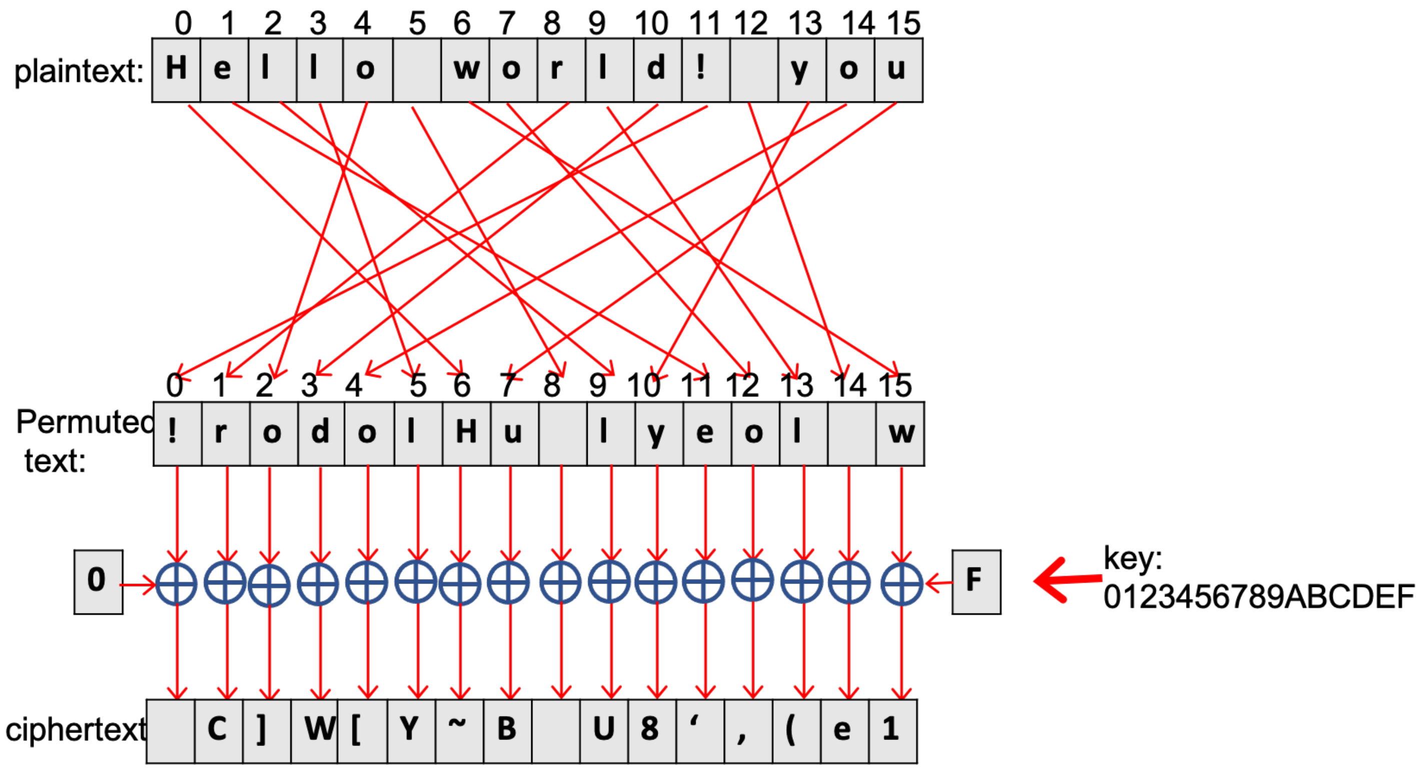
In this week's lab, you will build a symmetric cipher named **Derp**, using the C programming language. Derp is a block cipher of width 16 (ASCII characters) with key comprising 16 characters, as shown below.



Encryption with Derp is simple, permutation followed by key addition, as follows.



With 16 letters, there are 16! (=20922789888000) possible permutations. Derp uses one of them indicated in the image below. In the key addition layer, the secrete key is added to the permuted plaintext using the XOR operations (^ in C) applied to individual characters. For example, using "**Hello World! You**" and "**01234567890ABCDEF**" in place of plaintext and key, respectively, encryption works as follows:


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
plaintext: H e l l o w o r l d ! y o u
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Permuted text: ! r o d o l H u l y e o l w
0
F
key: 0123456789ABCDEF
ciphertext C] W [Y ~ B U 8 ' , (e 1

- In the diagram above, ⊕ denotes the XOR operation. Due to space limitation, only the first and last characters in the key are shown in the diagram. For example, the second character 'C' in ciphertext is obtained by xoring the plaintext letter 'r' and the key letter '1' as follows:

```
char x = 'r', y = '1';
char z = x ^ y;
printf("x ^ y = %c\n", z); // x ^ y = C

// z ^ x == y   and z ^ y == x
printf("z ^ x = %c\n", z ^ x); // z ^ x = 1
printf("z ^ y = %c\n", z ^ y); // z ^ y = r
```

- Note that there are several invisible ASCII characters. Your ciphertext may contain some of them.

Instructions

1. Starter code **derp.c** under the **Lab3** directory is provided for you to use.
2. Use the **`fgets`** function to prompt the user to enter plaintext.
   a. If the user enters less than 16 characters (without including the enter key at the end), you need to pad the string with additional characters, e.g. '_', to make it fit the Derp block size (Derp only works with strings of length 16.) If

the user types more than 16 characters, you will ignore the characters at index 16 or higher. If the length is exactly 16, you don't have to do padding or chopping. For simplicity, you may assume that the user input does not contain the underscore characters.

   b. Terminate your program if the user hits the enter key without entering any other characters.

3. Print the (padded) plaintext.
4. Encrypt it to obtain ciphertext and print the ciphertext.
5. Decrypt the ciphertext and print the recovered text.
6. Use the `strcmp` function to compare the recovered text and the (padded) plaintext, and print the result.
7. Go back to step 1.

Additional Requirements

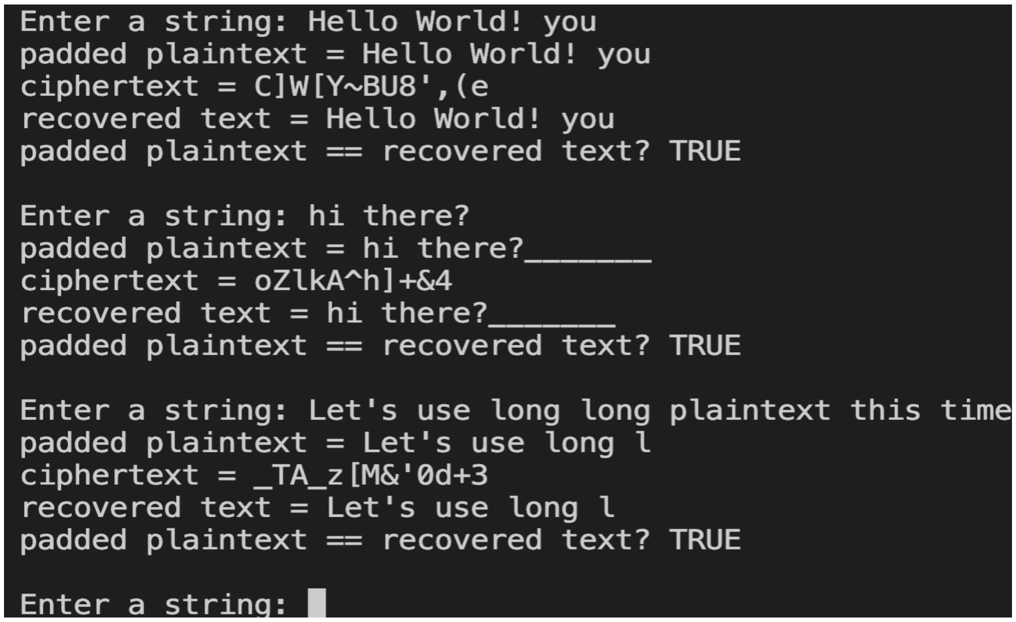
1. Please DRY (don't repeat yourself). You will not do this:

```
permuted[0] = plaintext[11];
permuted[1] = plaintext[8];
…
permuted[15] = plaintext[6];
```

2. In addition to the `main` function, you are required to write the following functions:
   - `padding`
   - `encrypt`
   - `decrypt`
   - `permutation`
   - `keyaddition`

   You may choose proper parameters and return type for each of the functions above. You are allowed to define one or more functions if needed.

Sample Output



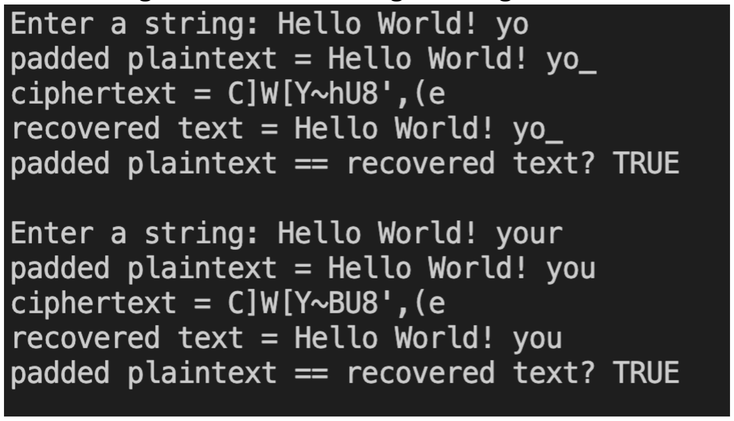
```
Enter a string: Hello World! you
padded plaintext = Hello World! you
ciphertext = C]W[Y~BU8',(e
recovered text = Hello World! you
padded plaintext == recovered text? TRUE

Enter a string: hi there?
padded plaintext = hi there?_______
ciphertext = oZlkA^h]+&4
recovered text = hi there?_______
padded plaintext == recovered text? TRUE

Enter a string: Let's use long long plaintext this time
padded plaintext = Let's use long l
ciphertext = _TA_z[M&'0d+3
recovered text = Let's use long l
padded plaintext == recovered text? TRUE

Enter a string: 
```

< More edges cases with strings of length 15 and 17>



```
Enter a string: Hello World! yo
padded plaintext = Hello World! yo_
ciphertext = C]W[Y~hU8',(e
recovered text = Hello World! yo_
padded plaintext == recovered text? TRUE

Enter a string: Hello World! your
padded plaintext = Hello World! you
ciphertext = C]W[Y~BU8',(e
recovered text = Hello World! you
padded plaintext == recovered text? TRUE
```

Deliverables:

1. You must upload your solution file **Lab3/derp.y** to your **Assignment 03** GitHub repository by 7 PM, 9/15/2021.
2. You must get sign-offs from one of the instructors or TAs during their office hours before the due date.

Your name:

Sign offs – Each signature is worth 1/N of your lab grade where N is the number of signatures

- The student used the `fgets` function and implement the `padding` function correctly.
- The student was able to use arrays to implement the permutation rule.
- Student could encrypt plaintext.
- The student could decrypt ciphertext.
- The student wrote all the required functions and used them properly.