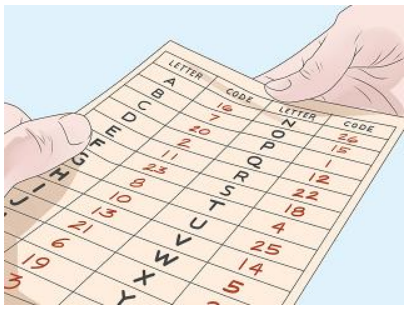
COMP2401 - Assignment #2

(Due: **Sun. Feb 9, 2020 @ 6pm**)

In this assignment, you will write two programs that involve string arrays. You will get used to comparing string characters as well as iterating through strings. You will also gain practice doing some bit manipulation.

---

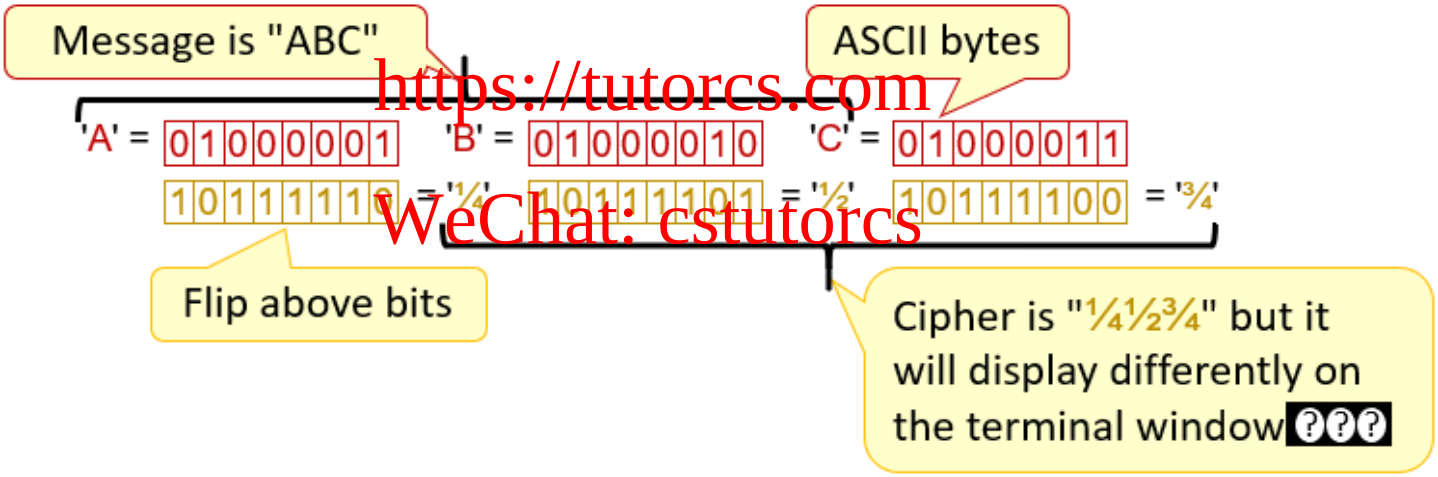
(1) The Bit-Flip Cipher



A ***cipher*** is a message written in secret code. We will implement some very basic ciphers as described below. Consider an original ***message*** **m[n]** made up of **n** characters that are found in a typical sentence (e.g., alphabetical/numeric characters, spaces, periods, commas etc..).

Write a program called **cipher1.c** that asks the user for an input string **m** (which is assumed to be 100 characters or less). It should then create a cipher by taking each character of **m** and flipping the bits. The result is the character that will be used in the cipher.

e.g., if m = "ABC", then the cipher is: "¼½¾". as follows:



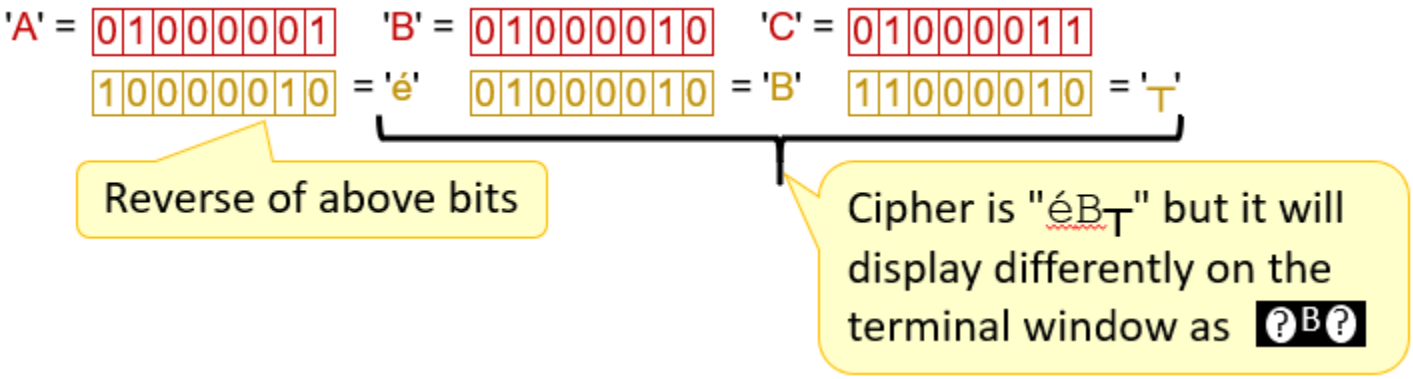
The code should create output that shows the binary representation of the characters and their ASCII codes and also shows the cipher characters in binary format and their ASCII codes. It should look as follows:

```
Enter the message to encode...
ABC

01000001 = 65
10111110 = -66

01000010 = 66
10111101 = -67

01000011 = 67
10111100 = -68

Cipher = "???"
```

(2) The Bit-Reverse Cipher

Copy your **cipher1.c** code to a file called **cipher2.c** and adjust **cipher2.c** so that it creates a cipher that takes each character of **m** and reverses the bits to get the cipher character.

e.g., if m = "ABC", then the cipher is: "éB┬" … as follows:



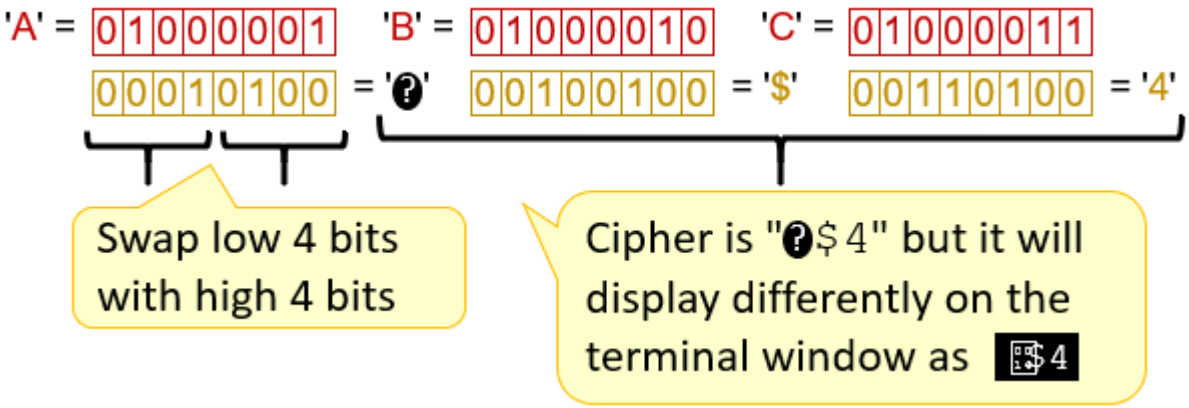
The code should create output that shows the binary representation of the characters and their ASCII codes and also show the cipher characters in binary format and their ASCII codes. It should look as follows:


```
Enter the message to encode...
ABC

01000001 = 65
10000010 = -126

01000010 = 66
01000010 = 66

01000011 = 67
11000010 = -62

Cipher = "?B?"
```


(3) The Nibble-Swap Cipher

Copy your **cipher2.c** code to a file called **cipher3.c** and adjust **cipher3.c** so that it creates a cipher that takes each character of **m** and swaps the low 4 bits with the high 4 bits to get the cipher character.

e.g., if m = "ABC", then the cipher is: "?$4", where ? is not displayable … as follows:



The code should create output that shows the binary representation of the characters and their ASCII codes and also shows the cipher characters in binary format and their ASCII codes. It should look as follows:

```
Enter the message to encode...
ABC

01000001 = 65
00010100 = 20

01000010 = 66
00100100 = 36

01000011 = 67
00110100 = 52

Cipher = "?$4"
```


(4) Random Cipher

Write a program called **cipher.c** that makes use of the three ciphers that you just created. Copy over your cipher-making code into this file. Adjust the main program so that it asks for the user message, and then randomly chooses one of the three ciphers and applies it to the message. The program should then output only (and exactly) 100 characters representing the encoded message followed by -1 valued bytes. If, for example, cipher1 is used and ABC is the message, then the program should output ¼½¾ followed by 97 bytes which are all -1. It is important that no other output is made to the screen (i.e., no other printf() calls … not even for asking for user input) because the next program will require exactly 100 bytes.

(5) Deciphering it all

Finally, write a program called **decipher.c** that will take the result of the **cipher.c** program and attempt to determine what the original message was by applying all three ciphers and seeing which one gives the most likely result. Now, although we cannot be sure which cipher was used, if we make some assumptions that the original text was mostly alphanumeric characters … then we can make a decent guess.

To approach this problem … first read in the characters from the standard input (i.e., getchar()). We will set up our testing so that the output from the cipher program is fed into the decipher program for testing (based on pipes … which we will discus later in the course).

Once the array of characters has been read in … you should try to decipher the message using each cipher and then choose the most likely match. To find the best match … you must take each deciphered character and determine if it is an alphanumeric character (i.e., is it an upper or lowercase letter or a digit from 0 to 9 … or the space character). You should count the percentage of decoded characters that are alphanumeric. The cipher that has the highest percentage of decoded alphanumeric characters … that will be the cipher that we will assume is the correct one and you should display that result. You should display the count for each cipher as well.

To test the code, use the following command (which assumes that the cipher and decipher programs are in the current directory):

```
./cipher | ./decipher
```

Here is what the output should look like:

```
student@COMP2401-F19:~$ ./cipher | ./decipher
This is an Encoded Message !
  Match =  46.43%  Deciphered String = ��i1��i1��y��l��9  ?Y??MY11yY?{
  Match =  96.43%  Deciphered String = This is an Encoded Message !
  Match =  57.14%  Deciphered String = ?ai?@i?@hg@*globjb@+j??hnj@H
Deciphered Message: "This is an Encoded Message !"
student@COMP2401-F19:~$
```


Make sure to test your code thoroughly.

IMPORTANT SUBMISSION INSTRUCTIONS:

Submit all of your **c source code** files as a single **tar** file containing:

1. A **Readme** text file containing
   - your name and studentNumber
   - a list of source files submitted
   - any specific instructions for compiling and/or running your code
2. All of your **.c source** files and all other files needed for testing/running your programs.
3. Any output files requested (e.g., **output.txt**).

The code **MUST** compile and run on the course VM.

- If your internet connection at home is down or does not work, we will not accept this as a reason for handing in an assignment late ... so make sure to submit the assignment WELL BEFORE it is due !

- You WILL lose marks on this assignment if any of your files are missing. So, make sure that you hand in the correct files and version of your assignment. You will also lose marks if your code is not **written neatly with proper indentation and containing a reasonable number of comments.** See course notes for examples of what is proper indentation, writing style and reasonable commenting).