```
-------------------------------------------------------------------------------
- INTRODUCTION                                                                -
-------------------------------------------------------------------------------
```

In this lab, you are going to implement an encoding algorithm which is used while
sending messages on network. The algorithm that you're going to use is called
"even bit-parity check" algorithm. It is used for whether a message received includes
any bit error, or not. It is very easy to understand, as follows:

--- Encoding Part ---
Assume there is a message that you want to send. It includes n bits consisting of
1 and 0's only. For each 8 bit of your message, you count the # of 1's. If there are
odd number of 1's then you add a 1 as the 9th bit of your message. If there are even
number of 1's then you add a 0 as the 9th bit of your message. This 9th bit is called
as parity bit. This is applied for each 8 bit along the message and in the end you
obtain a message of n + (n/8) bits (n will be a multiple of 8 in your lab, so don't
worry about it is divisible by 8 or not).

--- Decoding Part ---
When the receiver receives your message, it will also do the same calculation
and control whether the 9th bit (of each partial of 8 bits) is same with the one that
s/he calculated or not. If they are not the same, then it means there occured a bit
error while sending the message. It just understands that there is a bit error, but
it can't understand on which bit the error is (Actually, it is just capable of
determining the odd number of bit errors, and an insufficient algorithm when there
occured an even number of bit errors. However, we don't matter these things for the lab).
When it notices that there is an error, it declares the error. If there is not a
noticable error according to calculations, then it removes the parity bits and changes
the message into original version (the version without parity bits)).

```
-------------------------------------------------------------------------------
- LAB WORK                                                                    -
-------------------------------------------------------------------------------
```

Start your code by including a header named "lab3.h" as follows:

```
#include "lab3.h"
```

As was the case for the second in-lab exam, you are not going to define a "main"
function. Instead, you should define a function named "execute" which will be
called when you execute your program. Note that this function will not be used
by the automatic grader. It is provided only to enable you to test your
functions manually. In short, "execute" is your "main" function in the lab exam.

```
void execute();
```

In your source file, you are going to implement several functions whose
prototypes are present in the header file "lab3.h".

Warning: You need to provide empty definitions for all the functions even if you
do not want them to be graded. Otherwise, compiler gives an "undefined reference"

error. For instance, here follows an empty definition for execute:

```
void execute()
{
	/* code placed here will be executed when you click the "Execute" button */
}
```

ABOUT MEMORY USAGE: You are not allowed to allocate more space than is actually required. You may use "realloc" to claim space one-by-one (or by chunks of 8) if required. However, you are not supposed to allocate a very large space and then decrease the size with "realloc". You are allowed to reduce the size of a memory area at most 80 bytes.

In the lab system, each call to "malloc", "calloc" and "realloc" will be automatically investigated to check for extra memory usage. Therefore, you may not get a grade from a function if your memory usage is not validated.

The functions that you are going to write are explained together with examples in the document "lab3.h".