Introduction to C Programming in Unix Environment

Task 0: Maintaining a project using `make`

You should accomplish this task **before** attending the lab session. For this task, 3 files are provided: add.s, main.c, numbers.c.

1. Login into Linux.
2. Start up a text editor of your choice (vi, emacs, kate, pico, nano, femto, or whatever). It is **your responsibility** to know how to operate the text editor well enough for all tasks in all labs.
3. Using the above editor of your choice, write a Makefile for the given files (as explained in the introduction to GNU Make Manual, see the Reading Material and Lecture 1). The Makefile should provide targets for compiling the program and cleaning up the working directory.
4. Compile the project by executing make in the console
5. Obviously, you should read all the indicated reading material before attending the lab.

Task 1: Convert characters to bits

In this task you will develop a simple encoder that receives a sentence and outputs a "bar-code" to the screen. Download the code and implement the missing function in *bclib.c*.

NAME
encbc - text to binary.
SYNOPSIS
encbc [STRING] [STRING] …
DESCRIPTION
encbc converts the text given in the command line to its binary representation and outputs a bar-code that matches that representation, a column for each character where '#' matches 1 and ' ' (space) matches zero. Each character in ASCII has an integer representation, see ASCII table for more information. For example, the ASCII code of the 's' character is 115, which is 1110011 in binary representation.
EXAMPLES

```
$>./encbc shalom olam
# # ## # ##
#   #  #
   ### ## #
 # ### ## #
#
###########
###### ####
```

Mandatory requirements

- Implement the function ***'int* bc_char2bits(int ch, int *bits)'*** which receives as input a single character and an array of integers. The function will fill the array with the binary representation of the character and return it as an output. The function is located in the libbc.c file. *) Read the

bc_bits2char function and understand it before you start. Implement the *bc_char2bits* function in a similar manner. Read this tutorial (Bitwise operators section) to help you understand what << does. You can use this file to play with it.

- Make sure that the ***'bc_test()'*** function executes successfully without errors after implementing the ***'int* bc_char2bits(int ch, int *bits)'*** function.
- Create a makefile to compile your code, required in all tasks, even if it's not mentioned.

Notes

- BC_NBITS is the number of bits needed to represent a character.
- Bytes are represented in little-endian (from least significant to most significant bit).

Task 2: Build your own encoder

Implement the ***'encbc'*** program. For simplicity, we provide you with a bar-code decoder, i.e. the ***'decbc'*** program. This program receives a bar-code and returns its corresponding sentence. Here is an example that takes the output of the encoder and fed it to the decoder:

```
$ ./encbc hello | ./decbc
hello
```

Mandatory requirements

- Consult the ***'decbc'*** source code for implementation ideas.
- Test your program using *decbc* and show to the instructor that it works with some English sentences (would it work with other languages?).

Submission

- SUBMIT a ZIP file at the end of the lab session to the submission system .

Good luck!