Dalhousie University Faculty of Computer Science
CSCI 2122 Systems Programming Assignment 1

(1) In a computer system, memory operations currently take up 30% of execution time. A new gadget called a *cache* (*i.e.* an L1 cache) speeds-up 80% of the memory operations by a factor of 4.
   (a) **[3]** What is the speed-up due to the cache?
   (b) **[5]** A second new gadget called an L2 cache speeds-up *half the remaining 20% of the memory operations* by a factor of 2. What is the total speed-up with the L1 and L2 cache together?
   Note: It is useful to draw pictures to solve this part of the problem

(2) **[15]** Given the code stub in Figure 1 complete `main.c` using bitwise operations to provide the appropriate mapping between the two sets of ASCII characters in Table 2.
   - Compile the initial assignment code using:
     `gcc -o Assign01Code Assign01Code.c`
   - Assuming files containing the test text of Table 2, deploying the code from Figure 1 will produce an 'output' text file '`Out.txt`':
     `./Assign01Code <Alphabet.txt >Out.txt`
     or, the following will post the output directly to the screen:
     `./Assign01Code <Alphabet.txt`
   - A summary of the ASCII character set is available at:
     `https://en.wikipedia.org/wiki/ASCII`
   - *Hint:* For the two suggested input texts, use the table of ASCII codes to identify the bitwise mapping between input and outputs.
   - Your submitted code will be tested on a third text file to determine the validity of your solution.
   - *Important:* Leave the current code stub unchanged. The additional code should appear in the region indicated by the comment field.
   - Do not introduce other 'loop' structures

Input file (`Alphabet.txt`)	Output file (`Out.txt`)
`abcdefghijklmnopqrstuvwxyz`	`hpxaiqybjrzcks{dlt|emu}fnv`
`ABCDEFGHIJKLMNOPQRSTUVWXYZ`	`HPXAIQYBJRZCKS[DLT\EMU]FNV`

Table 1. Source (I/P) and Target (O/P) texts assuming ASCII encoded characters.

```
#include <stdio.h>

int main(void)
{
  int inChar, outChar;

  while ((inChar = getchar()) != EOF)
    {
     // Your bitwise code appears here…

     // last instruction in while loop
     putchar(outChar);
    }
  return 0;
}
```

FIGURE 1. Initial code snippet (Assign01Code.c). C library function `getchar(·)` returns an integer expressing the ASCII encoded character from the standard input (see note regarding redirection). The `putchar(·)` function converts the specified integer into the corresponding ASCII character and posts it to the standard output (again subject to any redirection). The only code that you need to add should fall between the two sets of comment characters.