DALHOUSIE UNIVERSITY
Inspiring Minds

Fall 2017 (Sep5-Dec5)
Faculty of Computer Science

CSCI 2132 — Software Development

Assignment 4

Due: *Thursday, Oct 19, 2017 by 11:59 p.m.*
Worth: 64 marks
Instructor: Vlado Keselj, CS bldg 432, 494-2893, vlado@dnlp.ca

---

Assignment Instructions:

Solutions to this assignment must be submitted through SVN, in a similar way as for the previous assignment.

The answers to all questions must be submitted in your SVN directory: *CSID*`/a4` where *CSID* is your CS userid. Remember that you need to add to `svn` the directory as well as any files that you want submitted.

1) (10 marks) If you finished the Dr. Tig's Git tutorial, you should have discovered his secret evil plan. Record all the parts of his plan in the plain textual file `a4q1.txt` and submit it as the answer to this question.

2) (24 marks) Anser the following question in a plain textual file named `a4q2.txt` and submit it. Since the question has several subquestions, you can copy the file `~prof2132/public/a4q2.txt` to your directory and edit this file further. This prepared file contains the questions to make it easier for you to type in the answers.

Let us assume that we have a computer with integers with size of 7 bits. We assume that integers are represented using 2's complement representation.

a) (4 marks) What would be signed and unsigned range of such integers?

b) (6 marks) What is bitwise representation of the following numbers as signed integers in this computer: 19, 36, -12, -17, -31, and -48.

Note: A bitwise representation means that you should show actual zeros and ones (7 digits) for each number.

c) (6 marks) Using bitwise representations from b), show in binary how to obtain results $36 + (-17) = 19$, $19 + (-31) = -12$, and $(-17) + (-31) = -48$

d) (5 marks) Using the results from c), find the result of $(-31) + (-48)$ in the above

representation? First find the bitwise representation, and then transform it to the actual number. Have in mind that we work only with 7 bit numbers.

e) (3 marks) Translate the binary floating point number 1101.01101 into the decimal floating point number. Show elements of your calculation.

3) (10 marks) Write a C program named `a4q3.c` that prints the sizes of several basic number types in the current architecture. Your program must use the `sizeof` operator. For example, the output of the program on bluenose should be:

```
Size of short       in bytes is:   2
Size of int         in bytes is:   4
Size of long        in bytes is:   8
Size of long long   in bytes is:   8
Size of float       in bytes is:   4
Size of double      in bytes is:   8
Size of long double in bytes is:  16
Size of char        in bytes is:   1
```

You should generate the output exactly as shown. The output file is also available as the file `~prof2132/public/a4q3-test.out` on bluenose.

Make sure that you program has an header comment describing the file, author, date, purpose, and description of the program, as shown in the template file `~prof2132/public/a4q3-template.c` on blunose. The content of this file is also shown here:

```
/* File:    a4q3.c
   Author:  Firstname Lastname, Banner_number
   Date:    Oct 13, 2017
   Purpose: Solution to problem A4Q3 of CSCI2132
   Description:  The program prints the size of various types using
     the sizeof operator.
*/
#include <stdio.h>

int main() {
  ...

  return 0;
}
```

4) (20 marks) Write a C program named `a4q4.c` that reads a sequence of integers, until the end of input, and prints their binary representation; i.e., their bitwise representation in memory.

Remember that if you are entering your input from the keyboard, you can indicate the end of input by pressing Ctrl-D. If you are using redirection to feed the standard input from a file, then you do not need to wory about indicating the end of input since it will be done by the system at the end of file.

Sample Input: An example of the input is:

```
0
1
-1
7
-7
1024
-1024
45560
-23470
```

(Sample Output) and for this input, the output file should be:

```
    0: 00000000 00000000 00000000 00000000
    1: 00000000 00000000 00000000 00000001
   -1: 11111111 11111111 11111111 11111111
    7: 00000000 00000000 00000000 00000111
   -7: 11111111 11111111 11111111 11111001
 1024: 00000000 00000000 00000100 00000000
-1024: 11111111 11111111 11111100 00000000
45560: 00000000 00000000 10110001 11111000
-23470: 11111111 11111111 10100100 01010010
```

You can notice that the bits are grouped in groups of 8 (per byte). Do not forget to add the header comment (as described in the previous question) to your program.

When writing the program, you may find helpful to have an `unsigned int` variable, which you can call mask that has one bit exactly set. You can achieve this by having the mask set to 1 and then shifting 1 to the left. If you are not familiar with the bitwise operators, you may find useful to read about them. In this particular solution, you may need '`<<`', '`<<=`', '`&`' '`>>`', and '`>>=`'.

In order to be aligned, the sample output prints the integers in 10 positions at the beginning of each line. You can find the sample input and output files for this question in the directory `~prof2132/public` on bluenose.