CSE1301 Exercise Sheet 2
Arithmetic and Boolean Expressions

(* Important to complete)

Exercise 1
Perform the indicated *bitwise* operations:

a) 10010111
AND 11101001
____

b) 11101010
AND 01010101
____

c) 01011100
AND 10100011
____

d) 10010111
OR 11101001
____

e) 11101010
OR 01010101
____

f) 01011100
OR 10100011
____

Note: In C, operator `&` performs bitwise AND, and the operator `|` performs bitwise OR

Exercise 2*
Check the precedence of the logical operators first, then fill in the following table to evaluate the C logical expression:

`x && y || x >= z && !z`

x	y	z				
0	0	0				
1	0	5				
0	1	1				
1	0	1				
-5	5	0				
0	5	-7				
5	0	0				
10	1	0				

Exercise 3*
Given the following logical expression, produce the truth table and draw the pictorial representation of the Boolean gates:

`w && x || !y || z`

Exercise 4
Evaluate the following C expressions:

(a) `10 - - 3 * 5`
(b) `-7 * - 3 - 7 % 3 - 7 / 3`
(c) `141 / 16 % 5`
(d) `141 % 16 / 5`

Exercise 5*
Indicate the order of evaluation in the following statement:

```
a = b = c - d % e * f && - - g / h + (i - j + k) ;
```

Exercise 6
Write a C expression that returns the value 1 if the integer variable year represents a leap year, and 0 otherwise (See Exercise Sheet 1, Exercise 3(b)).

Exercise 7

A group of `n` students is to be divided into 7 classes, as evenly as possible. (No class size should differ by more than 1 student when compared with any other class size.) Write a C expression for:

(a) the number of students in the smallest class
(b) the number of students in the largest class
(c) the average number of students per class;
(d) the number of classes of above average size;
(e) the number of classes of at most average size;
(f) the number of students in classes of larger than average size;
(g) the number of classes of exactly average size.

Additional Exercise: Cellular Automata

A (one-dimensional) *cellular automaton* can be represented by a list (or array) of *cells*. We consider a particular form of a CA in which each cell can be in one of 2 states: *alive* or *dead*. This array of cells is intended to evolve over time. Regard time as a series of steps numbered 1, 2, 3, ..., rather than a continuous quantity.

At each time, some cells will be alive and some dead, and it is necessary to determine which cells will be alive or dead at the next time (i.e., one time step later). This is done according to certain *rules*. The rules concern the states of a cell's *neighbours* (i.e., the cells next to it in the array), and take the following form (where *t* is the current time):

if a living cell has *i* or fewer living neighbours, then it dies
(i.e. becomes *dead* at the next time, *t* + 1) of starvation

else if a living cell has *j* or more living neighbours, then it dies of
overpopulation

else it must have > *i* neighbours but < *j* neighbours, and it stays alive at
the next time;

if a dead cell has fewer than *k* or more than *l* living neighbours,
then it stays dead

else it ``comes alive'' (i.e. the conditions are just right for life to start at
that cell; at the next time it is alive).

The rules are thus completely determined by choice of *i*, *j*, *k*, *l*, and these numbers stay fixed throughout the life of a particular array of cells. (We also need *i* < *j* and *k* < *l*.)

Write a program which reads in (a) *i*, *j*, *k*, *l*, (b) an initial array of cells, (c) the number of time steps over which the array is to evolve, and prints out what the array looks like at each time, with one line per time step, and with the lines properly aligned so it is easy for someone to see how the pattern of cells is changing over time.

You may like to generalise to 2-dimensional arrays of cells. For 2-D, the case *i = 1*, *j = 4*, *k = 2*, *l = 4* gives a particularly famous cellular automaton known as Life, due to J. H. Conway.

The study of cellular automata looks at how machines possessing computational capabilities and many of the characteristics of living organisms can be built from simple, identically-behaving parts. Cellular automata are also of interest in modelling biological systems and how such systems can develop computational ability.

References

A. W. Burks (ed.), *Essays on Cellular Automata*, University of Illinois Press, 1970.
M. Gardner, *Wheels, Life and Other Amusements*, W. H. Freeman & Co., 1983.