CpSc 418 **Homework 1** Due: Sept. 25, 2012, 11:59pm

50 points + 8 extra credit.
5% extra credit for problems 1–3 if solution submitted by 11:59pm on Sept. 20.

Please submit your solution using the handin program. Submit the program as
cs418 hw1
This requires you to have an account on the UBC Computer Science undergraduate machines. If you need an account, go to: `https://www.cs.ubc.ca/students/undergrad/services/account`
to request one.

Your submission should consist of the following two files:

`hw1.erl` – Erlang source (ASCII text).

`hw1.txt` – plain, ASCII text.

The first file, `hw1.erl`, will be your solution to the programming part of the assignment.
The second file, `hw1.txt`, will be a plain text file with your solution to the written (extra credit) part of the assignment (i.e. question 4).
No other file formats will be accepted.

1. **Bitonic Sequences (20 points).**

   (a) **Collapse (5 points).** Write an Erlang function
   `collapse(List) -> L`
   where `L` is a list of numbers that contains the elements of `List`, with adjacent duplicate elements collapsed into a single element. That is, if two or more adjacent elements of `List` have the same value, the generated list will only contain one of the elements of that value. For example:

   ```
   collapse([1,1,3,3,3,3,2]) -> [1,3,2].
   ```

   For this problem, you may not use any functions from the Erlang libraries (including the `lists` module).

   (b) **Bitonicity (15 points).** Write an Erlang function
   `is_bitonic(List) -> L`
   that returns `true` if its argument is a list of numbers that is a bitonic sequence. A bitonic sequence is one that contains a monotonically increasing sequence followed by a monotonically decreasing sequence, or vice-versa. A sequence is monotonically increasing if, for all elements `i` in the sequence, element(`i`) $\leq$ element(`i+1`). Monotonically decreasing is defined analogously. Note that all monotonic sequences are also bitonic sequences, as is the empty sequence. For example:

   ```
   is_bitonic([1, 3, 3, 5, 10, 1, 1, 0]) -> true;
   is_bitonic([1, 8, 7, 9]) -> false.
   ```

   For this problem, you may use any functions from the Erlang libraries (including the `lists` module).

2. **Arithmetic (30 points).**

   (a) **Decimal Addition (15 points).** Write an Erlang function
   `nsum(List1, List2) -> L`
   where `L` is a list of single-digit positive integers in base 10, and is the summation of the two arguments to the function, which are also single-digit positive integers in base 10. The elements of each such list represent digits of a number in base 10, such that the first (leftmost) element of the list represents the highest order digit, and the tail (rightmost) element of the list represents the lowest order ("ones") digit. For example, the number 506251 is represented by the list `[5,0,6,2,5,1]`.
   Numbers with leading zeros are accepted as input, but any output must contain no leading zeros. The only number that has 0 as its first digit is the number 0, which is uniquely represented as `[0]`. Furthermore, the empty list is acceptable as an argument to nsum and represents the number 0. The result of the function can be illustrated by the following examples:

```
nsum([1,5,9], [6,7,8]) -> [8,3,7];
nsum([0,6,9], [9,8,4]) -> [1,0,5,3].
```

For this problem, you may use any functions from the Erlang libraries. However, you may not just convert the lists an Erlang integer; perform the addition using the built-in Erlang + operator; and convert the result back to a list. Variations on such shortcuts will likewise receive zero or few marks.

(b) **Decimal Multiplication (15 points).** Write an Erlang function

```
nmul(List1, List2) -> L
```

where `L`, the arguments to the function, and the list semantics are as described in question 2a, but the result of the function is the product of the two numbers represented by the function arguments. For example:

```
nmul([3], [2,3]) -> [6,9];
nmul([2,1], [8,1]) -> [1,7,0,1].
```

You may use any functions from the Erlang libraries. As with `nsum`, you may not just convert the list to an Erlang integer, use Erlang's multiplication operator, and covert back to a list (or similar shortcuts).

3. **Processes (10 points).** Write an Erlang function

```
sum_first_n(N) -> integer()
```

that spawns `N` processes, $p_1, p_2, \ldots, p_N$, using the built-in spawn function, where process p_i will send a list containing two elements, namely $[i, i * 2]$, to the root process (i.e. the Erlang shell process). The root process will then receive the lists from all `N` processes and return the sum of all the integer elements contained in the spawned process' lists. For example:

```
sum_first_n(5) -> 45.
```

You may use any functions from the Erlang libraries.

4. **(8 points, extra credit).** For each problem on this assignment:

   (a) How long did it take you to solve the problem?

   (b) How long do you estimate that it would take you to solve a similar problem now that you have some Erlang programming experience?

   (c) Please rate each problem on a scale of 0 to 5 where

   0 – Worthless tedium.

   1 – Too much work, and I little learned.

   2 – A typical homework problem.

   3 – Definitely had a favorable learning/effort ratio.

   4 – I learned a lot and had fun doing so.

   5 – Wow! I've discovered a new way to think!