Assignment 2: Ada Programming (25%)

Choose **one** of the following topics.

1. SUDOKU

A Sudoku puzzle is a special kind of Latin square popularized by the Japanese puzzle company Nikoli in 1986, with Sudoku loosely translated to *single number*. Surprisingly, the game itself was designed by Howard Garns of Indiana, and published by Dell Magazines as *Number Place* in 1979. Latin squares, so named by the 18th century mathematician Leonard Euler (1707-1783), are $n \times n$ matrices that are filled with n symbols in such a way that the same symbol never appears twice in the same row or column. A standard Sudoku is a 9×9 grid with the additional property of having nine blocks (or subgrids) of 3×3 cells which contain the digits 1 to 9, with no repeat values allowed. This popular puzzle game demonstrates man versus machine, incorporating aspects of backtracking and recursion.

TASK

Design and implement an Ada program to solve Sudoku puzzles. You may use any technique you wish: back-tracking, recursion, concurrency, stacks, queues, lists, or add some human problem solving techniques. If using an algorithm which uses a data structure, please incorporate an Ada *package*. Test your puzzle with real Sudoku puzzles. Input should be in the form of a 9x9 Sudoku matrix stored in an ASCII file. A sample input might be of the form:

```
530070000
600195000
098000060
800060003
400803001
700020006
060000280
000419005
000080079
```

With zeros representing spaces. Output should take two forms: (i) output to standard output, and (ii) output to file. Output should be of the form:

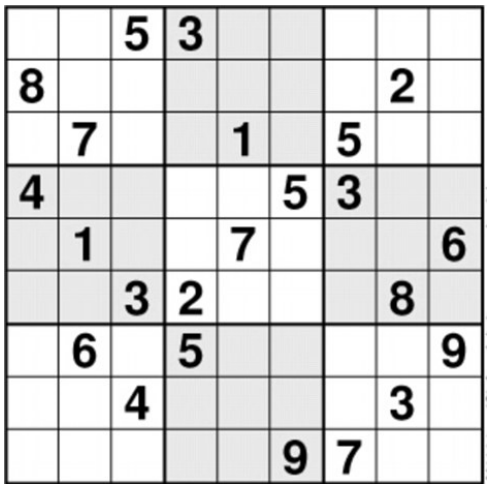
```
+-----+-----+-----+
|5 3 4|6 7 8|9 1 2|
|6 7 2|1 9 5|3 4 8|
|1 9 8|3 4 2|5 6 7|
+-----+-----+-----+
```

```
|8 5 9|7 6 1|4 2 3|
|4 2 6|8 5 3|7 9 1|
|7 1 3|9 2 4|8 5 6|
+-----+-----+-----+
|9 6 1|5 3 7|2 8 4|
|2 8 7|4 1 9|6 3 5|
|3 4 5|2 8 6|1 7 9|
+-----+-----+-----+
```

Note that the user should be prompted for both an input file and an output file (both of which should end in **`.txt`**). You may use an existing algorithm to solve the Sudoku puzzles, but make sure to reference any work you use.

TESTING

See if your solution can solve this puzzle, said to be amongst the hardest[1].



REFERENCES

- Crook, J.F., "A pencil-and-paper algorithm for solving Sudoku puzzles", *Notices of the AMS*, Vol. 56(4), pp.460-468 (2009).
- http://en.wikipedia.org/wiki/Sudoku_algorithms

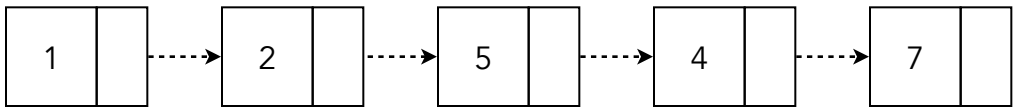
[1] http://www.dailymail.co.uk/news/article-1304222/It-took-months-create-long-crack–worlds-hardest-Sudoku.html

2. UNBOUNDED INTEGERS

The problem with integers is, that regardless of the system, they are bounded, making it difficult to calculate large factorials or higher Fibonacci numbers. Unbounded arithmetic operations are quite important in modern mathematical languages like Matlab, Mathematica, and Maple. In those systems unbounded signed integers are represented as *linked lists* of machine size integers. For example, consider the following series from the Factorial:

```
18 6402373705728000
19 121645100408832000
20 2432902008176640000
21 -4249290049419214848
```

When !21 is calculated, an overflow occurs. Using a linked list to store an integer, for example, the number 12547 would be represented as:



Adding two unbounded integers would require adding the associated linked list nodes in some manner.

TASK

Design and implement a Ada *package* to deal with unbounded integers. The package should store the numbers using a *linked list*, and allow standard operations such as +, -, *. Implement a user interface like the one shown in the next section to make it work. Remember, that you will need the usual functions to build a linked list, and print it out. In addition, use this package to implement a means of calculating any size Fibonacci number or Factorial, outputting it in its entirety. For example the value of 42! is:

```
1405006117752879898543142606244511569936384000000000
```

THE USER INTERFACE

The user interface will be very simple. It will be terminal input and output. For example:

```
> Enter an operation: + , - , * or !
 *
> Enter first operand:
-57890223456789999999999999999
> Enter second operand:
37894289377777777222222222222222
> The result is:
-219370887981581921271387585733497081501009602227777777777777778
```

ASSIGNMENT INFORMATION

DESIGN DOCUMENT

Discuss your program in 2-3 page *design document* (single-spaced), explaining your algorithm and decisions you made in the process of designing your program. Consider the design document a synopsis of your experience with Ada - this should be at least 1-2 pages in length. In addition, one page should include answers to the following questions:

- Would it have been easier to write the program in a language such as C?
- What were the greatest problems faced while designing the algorithm in Ada?
- Is there a better way of writing the program?
- What particular structures made Ada a good choice?
- Given your knowledge of programming, was Ada easy to learn?

DELIVERABLES

Either submission should consist of the following items:

- The design document.
- The code (well documented and styled appropriately of course).

SKILLS

- Ada programming, file I/O or Ada packages, recursion/data structures.

ASSIGNMENT INFORMATION

REFLECTION DOCUMENT

Discuss your program in 3 page (or more if you like) *reflection document* (single-spaced), explaining your algorithm and decisions you made in the process of designing your program. Consider the document a synopsis of your experience with Ada - this should be at least 2 pages in length. The write-up should explain your approach to the problem, including any modifications made to the specifications. In addition you should include answers to the following questions:

- What were the greatest problems faced while designing the algorithm in Ada?
- Was it easy to create a data structure, or perform recursion in Ada?
- What particular features made Ada a good language?

In addition, one page should include answers to the following questions:

- Would it have been easier to write the program in a language such as C?
- Is there a better way of writing the program?
- Given your knowledge of programming, was Ada easy to learn?
- What structures made Ada usable? (In comparison to C for instance)

DELIVERABLES

Either submission should consist of the following items:

- The reflection document.
- The code (well documented and styled appropriately of course).

SKILLS

- Ada programming, file I/O, recursion/data structures, packages.