CSC 225 SUMMER 2014
ALGORITHMS AND DATA STRUCTURES I
ASSIGNMENT 2
UNIVERSITY OF VICTORIA

1. Solve Problems 1.3.3 on Page 161 and 1.3.13 on Page 162 of the textbook.

2. Consider an implementation of a stack using an extendible array. That is, instead of giving up with a "StackFullException" when the stack becomes full, we replace the current array S of size N with a larger one of size $f(N)$ and continue processing the push operations. Suppose that we are given two possible choices to increase the size of the array: (1)$f(N) = N + c$ (for convenience, we start with an initial array of size 0) (2) $f(N) = 2N$ (we start with an initial array of size 1). Compare the two strategies and decide which one is better.

   To analyse the two choices, assume the following cost model: A "regular" push operation costs one unit of time. A "special" push operation, when the current stack is full, costs $f(N) + N + 1$ units of time. That is, we assume a cost of $f(N)$ units to create the new array, N units of time to copy the N elements and one unit of time to copy the new element.

3. Show the various steps of Selection Sort, Bubble Sort and Insertion Sort on the example array, $\{5, 7, 0, 3, 4, 2, 6, 1\}$.

4. In any array A, an *inversion* is a pair of entries that are out of order in A. That is, an inversion is a pair (i, j) such that $i < j$ and $A[i] > A[j]$. Develop a algorithm for computing the number of inversions in a given array. The running time of your algorithm should be $O(n + k)$ where k is the number of inversions in the input array.

5. Solve Problem 1.4.6 on Page 208 of the textbook.

6. An Array A contains $n - 1$ unique integers in the range $[0, n - 1]$. That is, there is one number in this range that is not in A. Describe in pseudo-code an $O(n)$-time algorithm for finding that number. You are only allowed to use $O(\log n)$ bits of additional space besides the array A itself.