University of Waterloo
CS240 - Spring 2017
Assignment 2

Due Date: Wednesday, May 31th, 5pm

Please refer to the course webpage for guidelines on submission. Submit your written solutions electronically as a PDF with file name `a02wp.pdf` using MarkUs. We will also accept individual question files named `a02q1w.pdf`, `a02q2w.pdf`, etc., if you wish to submit questions as you complete them.

Unless the base is explicitly specified, we assume in all questions that the logarithm operation is base 2.

Problem 1 [7+3=10 marks]

A sorting algorithm is said to sort *in place* if only a constant number of elements of the input are ever stored outside the array. In class we showed that all comparison based sorting algorithms require $\Omega(n \log n)$ comparisons to sort an array of length n. But suppose we are given an array $A[0 \ldots n-1]$ that contains a permutation of the first n non-negative integers. Allowing non-comparison based algorithms, give an $O(n)$ in place algorithm to sort A. Analyze the the running time of your method. *Note:* For simplicity we are assuming A is filled with integer keys. Your algorithm must easily extend to work for an array A that is filled with (key,element) pairs, each integer key in the range $0 \ldots n-1$ occurring exactly once.

Problem 2 [10 marks]

Suppose that each row of an n x n array A consists of 1s and 0s such that, in any row i of A, all the 1s come before any 0s in that row. Suppose further that the number of 1s in row i is at least the number in row $i+1$, for $i = 0, 1, \cdots, n-2$. Assuming A is already in memory, describe a method running in $O(n)$ time (not $O(n^2)$ time) for counting the number of 1s in the array A.

Problem 3 [7+5=12 marks]

Suppose you own n electronic devices, such as cell phones, tablets, portable music players, etc. Each of them comes with a charger cable, which you tossed into a box when you got it. But now it is time to recharge the devices, and so you must find for each one the correct charger cable.

Unfortunately the manufacturers didn't agree on a standard, and so while the charge cables look similar (they all have a small circular plug), the plugs have slightly different diameters, and so for each device only exactly one charger cable is correct. The differences

in diameters are small enough that you can't compare charger cables by themselves. The only thing you can do is to insert a plug into a device, which will tell you whether the plug fits, or is too big, or is too small.

Give a randomized algorithm that finds for all devices the matching charger cable and that uses $O(n \log n)$ operations of "try to put a plug into a device". You may use, without re-proving, any upper bounds on recursions that we have seen in class.

Bonus: Argue that any algorithm must use $\Omega(n \log n)$ such operations in the worst case.

Problem 4 [20 marks]

The enclosed file `binary.cc` contains a *C++* implementation of a priority queue using an array organized as binary heap. Recall from Assignment 1 that a *ternary* heap is like a binary heap except that nodes may have up to three children.

A *B-heap* (for some integer $B \geq 2$) is a heap where every node may have up to B children, every level except the last one is full, and the last level is filled from the left. It also satisfies the max-heap property: no node has a value that is bigger than the value of its parent. (Thus $B = 2$ gives a binary heap, and $B = 3$ gives a ternary heap.)

Implement a B-heap, where B is a parameter that is passed during the creation of the heap.

Some remarks:

- Your implementations must support the methods `MaxHeap(int B)`, `insert(int k)`, `deleteMax()`, `size()` and `levelOrderItem(int i)`.
- Do not change the class name or the headers of these methods.
- The run-time for these operations should be the same for B-heaps as it was for binary heaps (presuming B is a constant).
- The code contains some other methods (e.g. `printByLevel(int i, int k)`) which you might find helpful for testing purposes, but which we will not call in our testing scripts. You are allowed to add other methods as needed.
- You are allowed to code your own methods from scratch, rather than modifying the given implementation, as long as the methods give the same results.
- For testing purposes, your class should not have a main routine, or it should be surrounded by `#ifndef TESTING` and `#endif` (as done in the code).
- Submit a file `BHeap.cc`.
- Suggestion: We encourage you to implement a ternary heap first. This specific case should be quick to implement and will give you a good idea of what all needs to be done in the general case.