Midterm Examination Problem Sheet

TIME: 04/13/2013, 14:00–16:00

This is a open-book exam. You can use any printed materials as your reference during the exam. Any electronic devices are not allowed.
Any form of cheating, lying or plagiarism will not be tolerated. Students can get zero scores and/or get negative scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.
Both English and Chinese (if suited) are allowed for answering the questions. We do not accept any other languages.

There are 8 problems in the exam, each worth 25 points—the full credit is 200 points. Some problems come with a few sub-problems to help you get partial credits. For the 8 problems, 3 of them are marked with * and are supposedly simple; 3 of them are marked with ** and are supposedly regular; 2 of them are marked with *** and are supposedly difficult. There is one bonus problem, numbered 1126, with 10 points. The problems are roughly ordered by difficulty.

1. (25%, *) Considering representing a deque by a doubly-linked list, with an empty deque in the beginning. After executing each step of the following operations sequentially, list the contents of the doubly-linked list, with highlights on where the head and tail are.

   ```
   insertFront(1); insertBack(3); insertBack(2); insertFront(5); removeFront();
     removeBack(); insertBack(7); removeFront(); removeBack(); insertBack(6);
   ```

2. (25%, *) The following is a file system that can be represented by a tree rooted at `dsa`. Draw a diagram illustrating how the tree can be stored when the sub-trees are put in a linked-list-based container. You only need to draw links from parent to child as well as links between siblings, but not other links.

   ```
   dsa/instructor
   dsa/ta/ta1
   dsa/ta/ta2
   dsa/ta/ta3
   dsa/hw/hw1/write
   dsa/hw/hw1/prog
   dsa/hw/hw2/write
   dsa/hw/hw2/prog
   dsa/hw/hw2/data
   dsa/midterm/goodluck
   ```

3. (25%, *) Write down the outputs of the following C++ code.

   (a) (15%) Execute `max(arr, 0, 6, res)` with an array $arr = \{1, 2, 3, 4, 5, 6, 7\}$ and $res = 0$.

```
void max(int* arr, int left, int right, int& res){
  int mid = (left+right)/2;
  int res1 = 0;
  int res2 = 0;
  if (left == right){
    res = arr[mid];
  }
  else{
    max(arr, left, mid, res1);
    max(arr, mid+1, right, res2);
    std::cout << res1 << ' ' << res2 << ' ' << std::endl;
    res = (res1 > res2 ? res1 : res2);
  }
}
```

   (b) (10%) Execute `max(arr, 0, 6, res)` with an array $arr = \{1, 2, 3, 4, 5, 6, 7\}$ and $res = 0$.

```
void max(int* arr, int left, int right, int res){
  int mid = (left+right)/2;
  int res1 = 0;
  int res2 = 0;
  if (left == right){
    res = arr[mid];
  }
  else{
    max(arr, left, mid, res1);
    max(arr, mid+1, right, res2);
    std::cout << res1 << ' ' << res2 << ' ' << std::endl;
    res = (res1 > res2 ? res1 : res2);
  }
}
```

4. (25%, **) Consider having two stacks for storing integers, a `blue` one and a `green` one with the `push` and `pop` operations only.

   (a) (15%) Illustrate a scheme (with code or figure) that uses the two stacks and at most $O(1)$ of additional memory to simulate THREE stacks, numbered 1, 2, 3. In particular, illustrate how to implement the `push(int stack_number, int value)` and `pop(int stack_number)` operations clearly.

   (b) (10%) Illustrate a scheme (with code or figure) that uses the two stacks and at most $O(1)$ of not-in-stack memory to simulate K stacks, numbered 1, 2, 3, ..., K. Note that the amount of not-in-stack memory that you can use should be *independent* of K.

5. (25%, **) Consider mathematical k-dimensional vectors stored with a sparse array on the list. That is, a vector $\mathbf{v} = (3, 0, 0, 0, 9, 6)$ is stored as an ordered list of $(0, 3) \to (4, 9) \to (5, 6)$. Use any pseudo-code to describe an algorithm that computes the squared distance between two vectors $\mathbf{v}$ and $\mathbf{u}$. That is, $\sum_{i=0}^{k-1}(\mathbf{v}[i] - \mathbf{u}[i])^2$. The algorithm should run in time complexity linear to the number of non-zero terms in $\mathbf{v}$ and $\mathbf{u}$. Briefly justify how your algorithm achieves such a time complexity.

6. (25%, **) In Problems 6 and 7, the notation $f(n) = O(g(n))$ applies to functions $f(n)$ and $g(n)$ from $\mathbb{N}$ to $\mathbb{R}^+ \cup \{0\}$, and $f(n) = O(g(n))$ if and only if there exists $n_0 > 0$ and $c > 0$ such that $\forall n \geq n_0$, $f(n) \leq cg(n)$.

   (a) (15%) Assume that $k \in \mathbb{N}$. Let $f(n) = \log_2(1 + |a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} + \cdots + a_0|)$, and $g_k(n) = \log_2 n^k$. Prove that $f(n) = O(g_k(n))$. You can use the fact that $1 + |a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} + \cdots + a_0| = O(n^k)$, as well as the definition of $O(\cdot)$ above. But you cannot use any other theorems or claims beyond high-school math unless you prove them first.

   (b) (10%) Let $f(n) = \log_2(1 + |a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} + \cdots + a_0|)$, and $g(n) = \log_2 n$. Prove that $f(n) = O(g(n))$. You can use the result from the previous sub-problem (even if you don't prove it) if you want.

7. (25%, ***) Consider a non-decreasing function $h(m) \colon \mathbb{R}^+ \cup \{0\} \to \mathbb{R}^+ \cup \{0\}$, which means $h(m') \geq h(m)$ for all $m' \geq m$. Prove or disprove the following statement. "For functions $f(n)$ and $g(n)$ from $\mathbb{N}$ to $\mathbb{R}^+ \cup \{0\}$, if $f(n) = O(g(n))$, then $h(f(n)) = O(h(g(n)))$." (*Hint: this appears to be a more general statement that can be used to prove Problem 6(a). But is it still true?*)

8. (25%, ***) Consider a consecutive array of capacity N that holds k non-NULL elements, where $0 \leq k \leq N$. That is, all non-NULL elements are in positions 0, 1, 2, ..., $k - 1$, while all other positions contain NULL. Assume that you accidentally lose the value k and all you know is N and the array. Describe an $O(\log N)$-time algorithm that computes k. Briefly discuss why the algorithm is correct and how it achieves such a time complexity. (*Hint: think about binary search.*)

1126. (Bonus 10%, ???) You learned about selection sort in a very early stage of this class, and you learned about insertion sort in your reading assignment. Now, imagine that you are in the industry and your boss asks you to implement a sorting algorithm on an array by yourself—without using any of the existing implementations such as `qsort`. Would you choose selection sort or insertion sort? What is the reason that you'd use to *convince your boss that you are making a good choice*? (*Hint: Imagine that your boss, who is not of CS major, asks you this question. The TAs will grade qualitatively on whether your reason is correct and whether you can convince your boss in a reasonable manner.*)