CSE 373 Spring 2006

Data Structures and Algorithms

Midterm

Friday April 21st

NAME : _____

Do all your work on these pages. Do not add any pages. Use back pages if necessary. Show your work to get partial credit.

This exam is worth 50 points. After each question, you will find the number of points it is worth. You should spend approximately x minutes on a question worth x points.

1. 10 points
2. 15 points
3. 5 points
4. 14 points
5. 6 points

1. (10 points) Big-Oh and analysis of algorithms

(a) (2 points)
Let $f(n)$ and $g(n)$ be two functions. Define formally the meaning that $f(n)$ *is* $O(g(n))$

Let $f(n)$ and $g(n)$ be functions mapping nonnegative integers to real numbers. We say that $f(n)$ is $O(g(n)$ if there is a real constant c and an integer constant $n_0 \geq 1$ such that:

$$f(n) \leq c \times g(n) \ for \ n \geq n_0$$

(b) (4 points)

Which of the 4 functions below are (is) $O(n^2)$

- $f(n) = n^2/(6n+1)$
- $g(n) = n^2 \times (n+1)$
- $h(n) = n^2 \times logn$
- $i(n) = 2^{2logn}$

$f(n)$ and $i(n)$

(c) (2 points)

Given an array A of n elements, the program (pseudo-code) below returns an array B such that $B(i) = \sum_{j=0}^{i} A(j)$

```
integer s;
s := 0;
for (i:=0, i<n: i++) {
  s := s + A[i];
  B[i] := s;
}
```

What is the time complexity of this program?

$O(n)$

(d) (2 points)

If $A(i) = i$, $i = 0, 1, ..., n-1$ what is the value of $B(i)$?

$$B(i) = \sum_{j=0}^{i} j = \frac{i \times (i+1)}{2}$$

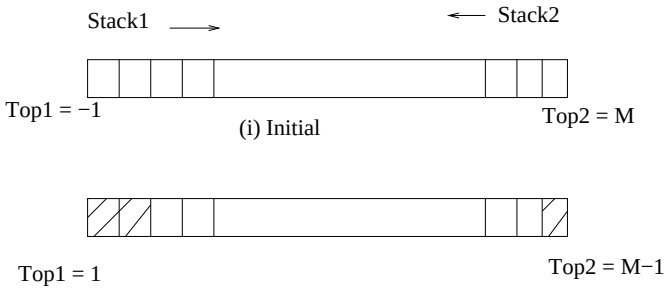
2. (15 points) Memory allocation (This question continues on the next page)

A block of memory of size M is used by a memory allocator. All blocks that are allocated and deallocated are of the same size, that is unit size. In other words a maximum of M blocks can be allocated at any given time.

The memory allocator sees the block of memory as an array of M elements, from element 0 to element $M-1$.

(a) (5 points)

The allocations/deallocations are only for 2 stacks, *Stack1* growing from block 0 up toward block $M-1$ and *Stack2* growing from block $M-1$ down to block 0. Let $Top1$ (initialized to -1) and $Top2$ (initialized to M) be the tops of the stacks. For example, after 2 calls to *AllocStack1(Top1,Top2)* that allocates elements in *Stack1* followed by 1 call to *AllocStack2(Top1,Top2)* that allocates elements in *Stack2*, the memory that was initially all free blocks now has 3 allocated blocks as per the figure:



Give pseudo-code for the 2 methods: *AllocStack1(Top1,Top2)* and *AllocStack2(Top1,Top2)* that return the index (a value between 0 and $(M-1)$ of the block that was allocated. Be sure to include a test for overflow.

```
  AllocateStack1(integer Top1, integer Top2): integer {
    Top1++;
    if (Top1 = Top2) then throw overflow exception
      else return Top1
  }
AllocateStack2(integer Top1, integer Top2) : integer {
    Top2--;
    if (Top1 = Top2) then throw overflow exception
      else return Top2
  }
```

(b) (5 points)

If now *Stack1* was replaced by a queue with *Front* and *Rear* pointers, what modification would you make to the method to allocate in the stack (i.e., *Stack2*). (You don't have to write the pseudo-code; just state what would change and how it would change).

I would just change the test for overflow to compare Top2 and Rear.

(c) (3 points)

Could you have a case in part (b) where you cannot allocate for the stack but there are some free blocks in the memory? If your answer is yes what is the worst case, i.e., the minimum number of elements in the queue for which this could happen? Justify your answers in 1 or 2 sentences.

Yes. Consider a few allocations in the queue (without overflow) followed by a deallocation. If now an allocation in Stack2 results in an overflow, there is a free block (block at index 0). In fact, the worst case is when the queue has only 1 element.

(d) (2 points)

In order to remedy the problems you found in part (c), a friend of yours says that you should use a circular queue implementation. Would that help? Justify your answer in 1 or 2 sentences.

No. The situation above could still occur

3. (5 points) Singly Linked List implementation

You are given a singly linked list with a pointer to its first node, called *head*, a pointer to its last node, called *tail*, and a pointer to a node in between called p. You can assume that the list has at least 3 nodes.

(a) (3 points)

Give pseudo-code to transform the above list into a list which is the concatenation of the list starting at p (included) and terminating at *tail* (included) with the list starting at *head* (included) and terminating at p (not included). The method can destruct the original list. Only a pointer to the first node of the new list is required.

```
   Splitandconcatenate(p,head,tail: pointer): pointer{
     curr : pointer;
     tail.next := head;
     curr := head;
     while curr.next != p {
       curr := curr.next;
      }
     curr.next := null
     return p
}
```

(b) (1 point)

What is the time complexity of your pseudo-code above if you know that the list has n elements and that p is the node directly following the *head* node.

$O(1)$

(c) (1 point)

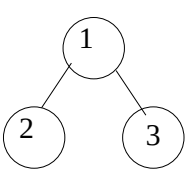
What is the time complexity of your pseudo-code above if you know that the list has n elements and that p is the node directly preceding the *tail* node.

$O(n)$

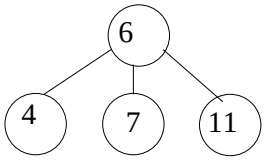
4. (14 points) Binary Search Trees (This question continues on the next page)

(a) (2 points)

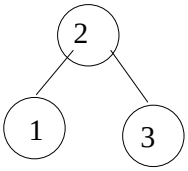
Which of the following 4 trees are Binary Search Trees (BSTs)



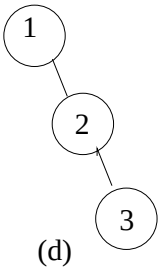
(a)



(b)



(c)

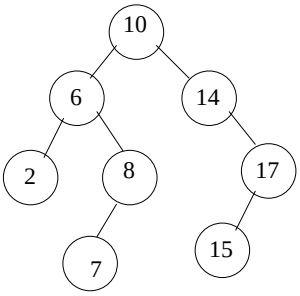


(d)

(c) and (d)

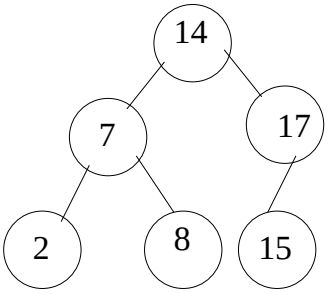
(b) (3 points)

Draw the BST where the following keys are inserted in the order:
10, 6, 14, 17, 15, 8, 2, 7



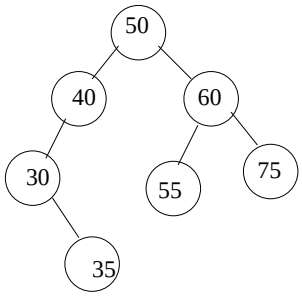
(c) (3 points)

Given the BST of part (b) draw the BST after the deletion of key 10 followed by the deletion of key 6



(d) (3 points)
Recall that a traversal in:
preorder means traverse root, traverse left subtree, traverse right subtree
postorder means traverse left subtree, traverse right subtree, traverse root
inorder means traverse left subtree, traverse root, traverse right subtree

Given the postorder of a BST: 35, 30, 40, 55, 75, 60, 50
draw the tree



(e) (3 points)
Give the pseudo-code for a method Findmax() that returns the maximum element in a BST.

```
   Findmax(T: tree pointer) : integer {
    if T = null then throw exception empty tree;
    while (T.right != null) {
      T := T.right;
     }
    return T.data;
}
```

5. (6 points)

Fill up the table below with the time complexity in Big-Oh notation for the average cases and worst cases of 3 sorting algorithms

Sorting Algorithm	Average Case	Worst Case
Insertion Sort	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$
MergeSort	$O(nlogn)$	$O(nlogn)$