CSE373 Midterm I

Fall 2009

(Closed book, closed notes)

Name: .....

Problem	Points
Q1 [20 points] Q2 [15 points] Q3 [15 points] Q4 [10 points] Q5 [15 points] Q6 [25 points]	
Total [100 points]	

1 [20 points] Asymptotic Complexity

Indicate the running time for the following functions. You MUST choose your answer from the following list (not given in any particular order), each of which could be re-used (could be the answer for more than one of the functions below):

$$O(n), O(n^2), O(n^3), O(n^4), O(n^5), O(\left(\frac{1+\sqrt{5}}{2}\right)^n), O(2^n), O(n!),$$

$$O(\log n), O(\log^2 n), O(n \log n), O(n^{\frac{3}{2}})$$

1.
```
int f1(int n, int a[]) {
   s = 0;
   for (int i = 0; i < n; ++i) {
       for (int j = 0; j < n; ++j) {
          s = s + abs(a[i] - a[j]);
       }
   }
return s;
}
```

Answer: $\underline{O(n^2)}$

2.
```
int f2(int n, int a[]) {
   s = 0;
   for (int i = 0; i < n; ++i) {
       if (a[i]>n)
         for (int j = 0; j < i; ++j) {
           s = s + a[i]*a[j];
       }
   }
return s;
}
```

Answer: $\underline{O(n^2)}$

3.
```
int f3(int n {
   s = 0;
   for (int i = 0; i < n; ++i) {
      for (int j = 0; j < i*i; ++j) {
         s = s + j;
       }
   }
return s;
}
```

Answer: $\underline{O(n^3)}$

4.
```
int f4(int n) {
  if (n > 1)
     return 1 + 2*f4(n-1);
  return 0;
}
```

Answer: $\underline{O(n)}$

5.
```
int f5(int n) {
   if (n > 1)
      return 3*f5(n/2) + 1
   return 0;
}
```

Answer: $\underline{O(\log n)}$

6.
```
int f6(int n) {
   if (n > 1)
      return 3*f6(n-1) + 2*f6(n-2);
   return 1;
}
```

Answer: $\underline{O(\left(\frac{1+\sqrt{5}}{2}\right)^n)}$

7.
```
int f7(int n) {
   s = 1;
   for (i=0; i<n; i++)
     s = s + f7(i);
   return s;
}
```

Answer: $\underline{O(2^n)}$

8.
```
int f8(int n, int a[]) {
   s = 1;
   for (i=0; i<n; i++)
     s = s + a[n-1]*a[i]*f8(n-1,a);
   return s;
}
```

Answer: $\underline{O(n!)}$

2 [15 points] Algorithm Design

Consider the functions below; each has its asymptotic running time indicated right before it. For each function, write another function that computes the same value and has a better asymptotic running time, as indicated in each case. Hint: you need to figure out what the function computes, then ask yourself if you can compute the same thing more efficiently.

1. The following function runs in time $O(n)$:

```
int g1(int n, int a[]) {
   s = 0;
   for (int i = 1; i < n; ++i)
       s = s + a[i] - a[i-1];
return s;
}
```

   Write an equivalent function that runs in time $O(1)$.

   Answer:

```
int g1(int n, int a[]) {
  return a[n-1]-a[0];
}
```

2. The following function runs in time $O(n^2)$:

```
int g2(int n, int a[]) {
   s = 0;
   for (int i = 0; i < n; ++i) {
       for (int j = 0; j < n; ++j) {
          s = s + a[i]*a[j];
       }
   }
return s;
}
```

   Write an equivalent function that runs in time $O(n)$.

   Answer:

```
int g2(int n, int a[]) {
     int s=0;
     for(int i = 0; i < n; ++i) {
        s += a[i];
     }
     return s*s;
}
```

3. The following function runs in time $O(n^2)$:

```
int g3(int n, int a[]) {
   s = 0;
   for (int i = 0; i < n; ++i) {
       for (int j = 0; j < i; ++j) {
          s = s + a[i]*a[j];
       }
   }
return s;
}
```

Write an equivalent function that runs in time $O(n)$.

Answer:

```
int g3(int n, int a[]) {
     int s=0;
     int sumSquares = 0;
     for(int i =0; i<n; ++i) {
       s += a[i];
       sumSquares += (a[i]*a[i]);
     }
     return ((s*s)-sumSquares)/2;
}
```

3 [15 points] Big O Notation

1. For each of the functions below, give the simplest, tight big-O expression. For example, if the function where $n + 20$, then you should answer $O(n)$:

$$\begin{aligned} n^{1/3} + n^{1/4} + \log n &= O(n^{1/3}) \\ \log n^2 + \log^2 n &= O((\log n)^2) \\ 2^{3n} + 3^{2n} &= O(9^n) \\ n! + 2^n &= O(n!) \\ n^2 + 2^n &= O(2^n) \end{aligned}$$

2. For each of the statements below indicate whether it is true of false. You don't have to justify your answer.

   (a) If $f(n) = O(g(n))$ then $g(n) = \Omega(f(n))$.

   Answer: True: X False:

   (b) If $f(n) = O(g(n))$ then $f(n) = o(f(n))$.

   Answer: True: False: X

   (c) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$ then $f(n) = O(h(n))$.

   Answer: True: X False:

   (d) If $f(n) = O(g(n))$ then for every n, $f(n) \leq g(n)$.

   Answer: True: False: X

   (e) If $f(n) = O(g(n))$ then $f(n) + g(n) = O(g(n))$.

   Answer: True: X False:

4 [10 points] Stacks and Queues

The array a contains the following 3 elements:

a = | a | b | c |

For each of the following two program fragments indicate what they print:

```
n = a.size();
Stack s = Stack();
for (i=0; i<n; i++) {
    s.push(a[i]);
    for (j=0; j<i; j++)
       s.push(a[j]);
    }
while (!s.empty()) {
    System.out.println(s.pop());
}
```

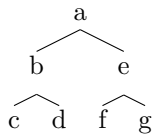
Answer: b a c a b a

```
n = a.size();
Queue s = Queue();
for (i=0; i<n; i++) {
    s.equeue(a[i]);
    for (j=0; j<i; j++)
       s.enqueue(a[j]);
    }
while (!s.empty()) {
    System.out.println(s.dequeue());
}
```

Answer: a b a c a b

5 [15 points] Tree Traversals

Consider the tree below:



Show the output for each of functions below:

1.
```
void t1(Node x) {
    if (x != null) {
       System.out.print(x.element);
       t1(x.left);
       t1(x.right);
       System.out.print(x.element);
   }
```

Answer: `abccddbeffggea`

2.
```
void t2(Node x) {
    if (x != null) {
       System.out.print(x.element);
       t2(x.right);
       t2(x.left);
   }
  }
```

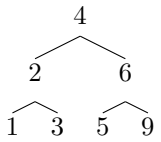
Answer: `aegfbdc`

3.
```
void t3(Node x) {
    if (x != null) {
       t3(x.left);
       t3(x.right);
       System.out.print(x.element);
       t3(x.right);
       t3(x.left);
   }
  }
```

Answer: `cdbdcfgegfafgegfcdbdc`

6 [25 points] Search Trees

1. Show the result of inserting 2, 1, 4, 5, 9, 3, 6 into an initially empty AVL tree. Showing the final result is good enough to get full credit, but showing the process might get partial credits.



2. For each of the statements below indicate whether it is true or false in all AVL trees that have at least two nodes:

   (a) The element at the root node is the median element in the tree.

   Answer: True: False: X

   (b) The minimum element in the tree is on a leaf node.

   Answer: True: False: X

   (c) The node that stores the second smallest element has a left child.

   Answer: True: False: X

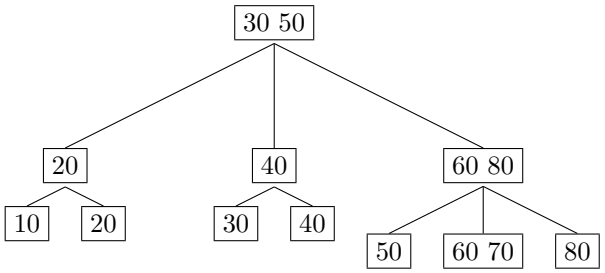
(d) The depths of any two leaf nodes differs by no more than two.

Answer: True: False: X

(e) If we traverse the tree in-order, then the we visit the elements in their increasing order.

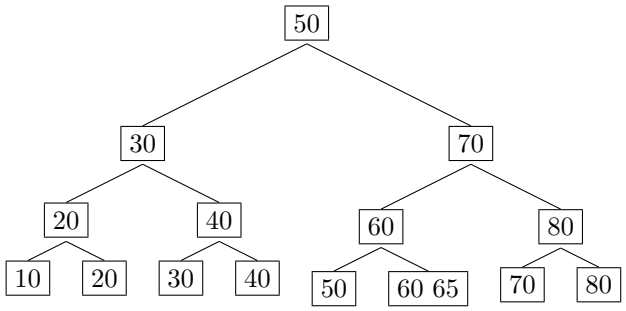
Answer: True: X False:

3. Consider the B tree below, where $M = 3$ and $L = 2$.



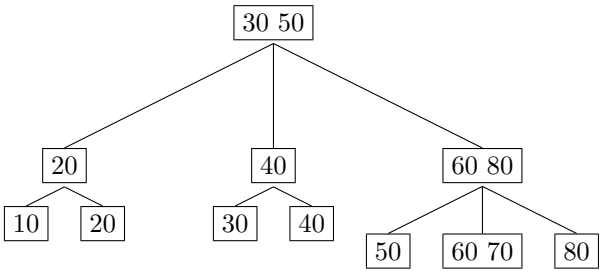
(a) Show the B tree obtained after inserting the element 65. You have to show one B tree.

Answer:



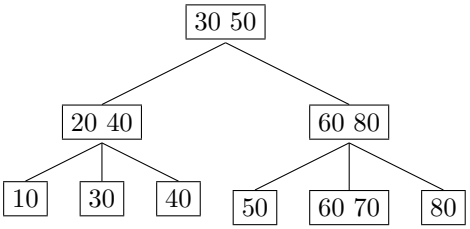
Also correct is to split the overflow node 60 65 70 into 60 and 65 70.

(b) Consider again the original tree, shown here again, for your convenience:



Show the B tree obtained after deleting the elements 20 and 30, in this order. You have two show two B trees; you may also show the process for possible partial credits.

Answer:



Alternatively, the key 20 in the node [20 40] can be replaced by 30.