Fall 2003 CS61B Midterm (50/300 points)

;;;;
Meta
;;;;

GS = Grading Standard

We've tried to include the common errors and grading standard for every question.

;;;;;;;;;;
QUESTION 1
;;;;;;;;;;

GS: The T/F questions were worth 1/2 point each. You got a -1/2 point if you answered a T/F question incorrectly. The fill-in-the-blank questions were worth 1 point each.

a) F

b) F

c) T

d) T

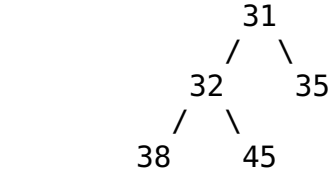
e) 4^h

f) 3

;;;;;;;;;;
QUESTION 2
;;;;;;;;;;

GS: 5 points total, but everyone received full credit for parts d) and e).

a) 1 point for drawing the correct heap after removing 20



b) 1/2 point

| 20 | 31 | 35 | 38 | 32 | 45 |

c) 1/2 point

```
 -------------------------------
 | 31 | 32 | 35 | 38 | 45 |    |
 -------------------------------
```

d) 1.5 points

   This question was impossible to answer because binary heaps
   must be complete. With the numbers provided, you could not
   form a complete binary search tree, thus you couldn't form a
   max binary heap. Everyone received all the points for this question.

e) 1.5 points

   This question was thrown out as well because of the same reason as
   (d). It could be impossible to make both a balanced BST and a max
   heap from a given set of data. Everyone received the max points for it.

;;;;;;;;;;
QUESTION 3
;;;;;;;;;;

GS: 4 points

Each part was worth 1/2 point.

a) >
b) >
c) =
d) =
e) =
f) <
g) =
h) =

;;;;;;;;;;
QUESTION 4
;;;;;;;;;;

GS: 2 points, 1 point for each part.

a) For any number n, all numbers n + 5*i where i is a positive integer >= 0
   will be mapped to the same location as n.
   For example: (5, 10, 15) or (7, 12, 17)

b) g(n) is better because f(n) can't ever map to an even number. f(n) effectiv
   reduces the possible buckets by half.

;;;;;;;;;;
QUESTION 5
;;;;;;;;;;

GS: Part a) was worth 2 points with each blank being worth 1/2 point. If you forgot to put object()'s constructor call, you were only penalized 1/2 poin one time.

Part b) was worth 4 points total. 1 point was taken off for each incorrect blank up to 0 points total for the question.

a) i. Dog, Animal, Object
ii. Rabbit, Animal, Object
iii. Poodle, Dog, Animal, Object
iv. None

Common errors: Almost everyone forgot to put the Object constructor. There we a handful of people that caught this though.

b) i. Animal
ii. Rabbit
iii. Dog
iv. Dog
v. Rabbit
vi. CT

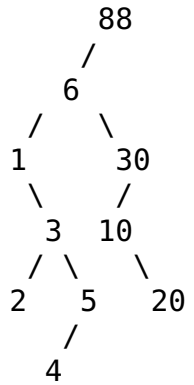
Common errors: Most people didn't know that vi. will cause a Compile-time exce This happens because Animal doesn't define a hop() method and thus you must cas the variable b to a Rabbit before you can call that method on it.
i.e. ((Rabbit)b).hop();

;;;;;;;;;;
QUESTION 6
;;;;;;;;;;

GS: Part a) was worth 3 points. You either got the entire 3 points or nothing. For part b), you were given the full 1 point if you wrote out the in-order traversal for the tree you drew in part a).

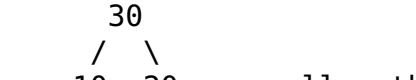
Part c) was worth 1 point with no partial credit.

a)



Common errors: A lot of people didn't verify that their tree was in fact a BST By looking at your tree, you should never have a larger number to the left of

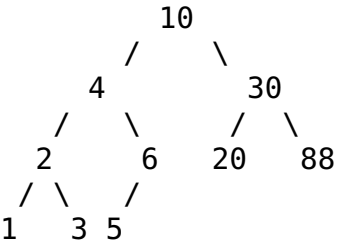
a smaller number and vice-versa. An examples of what people put is:



b) 1, 2, 3, 4, 5, 6, 10, 20, 30, 88

The in-order traversal of a BST are the numbers from lowest to highest. But to get full credit for this problem, you must put the in-order traversal of the tree you drew in part a). So if you drew the wrong tree, you must put the in-order traversal of that tree. If you drew the wrong tree, but gave the in-order traversal for the correct tree, we only took off 1/2 point.

c)



Common errors: A lot of people drew a minimal height tree, but didn't make it *complete*.

;;;;;;;;;;
QUESTION 7
;;;;;;;;;;

GS: 8 points. The grading for this is similar to question 3 on the Quiz.

```
public Object dequeue() throws EmptyQueueException {

  if( empty() ) throw new EmptyQueueException();

  Object o = pop();
  if( empty() ) return o;

  Object front = this.dequeue();
  enqueue( o );
  return front;
}
```

* -1/2 pt for the first instance of a small compile-time error

Examples:

- Omitting parenthesis in function calls (eg dequeue vs dequeue()... only deducted for the first time).
- Misspelled methods

* -1 for first instance of severe compile-time errors

Examples:

- Forgetting to return a value (if you declared the function non-void)
- Not declaraing variables (such as the Objects o and front)
- Calling dequeue recursively and passing it 'this' or something else despite
  the fact that it takes no parameters

* -1 for all instances of run-time errors/statements that generate
  the wrong output.

;;;;;;;;;;
QUESTION 8
;;;;;;;;;;

GS: 16 points total

The answers for part a) and b) below are broken up into 4 columns: the # points
using that datastructure, and the properties that must hold to use that ds and
For each of the three methods, we've outlined the credit you would receive give
the solutions that used the most appropriate datastructures in a way that maxim
for that method.

For example, if I put the following answer for lookupByName():

Method	DS	Worst Case Efficiency	Properties
lookupByName()	3	O(log n)	Each student is put in

Then by looking at the following table, I see that 2 points were awarded for th
efficient to use a Hashtable with the student name, a string object, as the key
time would be O(1) as opposed to O(log n) with a BBST.

There was a constraint added during the exam that your nextElement() method of
run in 0(1) time.

Problem 8a

# pts given	datastructure	big-O	properties
sortByScore			
3	Vector (1)	O(1)	Maintain in sorted orde
?		O(n)	Need to clone to preven
0		O(n log n)	Unsorted
0		O(n^2)	Unsorted
3	Linked List (2)	O(1)	Sorted
?		O(n)	Need to clone to preven
0		O(n log n)	Unsorted
0		O(n^2)	Unsorted

```
        3       Balanced tree (3)       O(n)            Do an inorder traversal
        1                               O(log n)        Inorder traversal done
        1                               O(log n)        Explain that it is O(lo
        0.5                             O(log n)        Don't say that you need
        0       Hashtable (4)           *               *
        0       Stack (5)               *               *
        0       Queue (6)               *               *
        2       Heap (7)                O(n log n)      Copy heap, O(n), remove
        1                               O(n)            Copy heap, and do heap
        0                               O(1)            Forget to copy, enumera
        ?                               O(1)            nextElement moves from
        0                               O(log n)        *

lookupByName
        3       Hash Table (4)          O(1)            * Anything true
        3                               O(n)            Bad hash code (but it's
        1                               O(n)
        2       Balanced Tree (3)       O(log n)
        3       Vector (1)              O(1)            Write your own hash tab
        2       Vector (1)              O(log n)        Sorted, Binary search
        0       Others (2,5-7)          *               *

studentsWithScoreAtLeast
        3       Vector (1)              O(1)            Maintain in sorted orde
        1                               O(n)            Sorted with linear sear
        1                               O(n)            Unsorted, copy elements
        1                               O(log n)        Binary search in first
        0                               O(n log n)      Unsorted
        0                               O(n^2)          Unsorted
        3       Linked List (2)         O(1)            Sorted in descending or
        1                               O(n)            Sorted with linear sear
        1                               O(n)            Unsorted, copy elements
        0                               *               Unsorted
        1       Balanced tree (3)       O(n)            Do an inorder traversal
        3                               O(log n)        Walk down to starting e
        1                               O(log n)        Explain that it is O(lo
        0.5                             O(log n)        Insufficient explanatio
        0       Hashtable (4)           *               *
        1       Stack (5)               O(n)            Pop all off, push all b
        0       Stack (5)               O(n)            Insufficient explanatio
        1       Queue (6)               O(n)            Dequeue all, enqueue al
        0       Queue (6)               O(n)            Insufficient explanatio
        0       Heap (7)                *               *  (A heap can only be

Problem 8b
lookupByLogin
        3       Hash Table (4)          O(1)            Only if you didn't use
        3       Vector (1)              O(1)            Direct mapped table: Us
        2       Balanced Tree (3)       O(log n)
        2       Vector (1)              O(log n)        Sorted vector with bina
        0       Others (2,5-7)          *               *
```

For part c), the worst case efficiency is specified below depending on the data

Problem 8c

1	O(n)	If sorted vector or if
1	O(log n)	If BBST

For part d), some people didn't realize that although you could only use 1 type
each instance for a different purpose. For example, if I chose a sorted vector
I sort on and make another instance with the student's score as the key. Then
and O(1) for studentsWithScoreAtLeast. Only a sorted vector and BBST provided

Problem 8d

# pts	datastructure	reasoning
3	Sorted Vector (1)	Can do all lookups in O
3	Balanced BST (3)	Can do all lookups in O
0	*	Makes no logical sense