

COMP182 Final Review - Solution Sketches

Problem 1. (10 points) Assume that you have an implementation of a linked list using nodes. Write two methods for this class. First, write a method which returns the number of items in the linked list. Second write a method which deletes the first half (rounded down) of the linked list (eg if the list contains 5 nodes then delete the first two items).

```
public int numberOfItems() {
    int count=0;
    Node nd = head;
    while (nd != null) {
        count++;
        nd = nd.getNext();
    }
    return count;
}

public void deleteFirstHalf() {
    Node nd = head;
    for (int i=0; i<numberOfItems(); i++)
        nd = nd.getNext();
    head = nd;
}
```

Problem 2. (10 points) Give the number of basic steps executed by calling the method below (as a function of n). What is its time complexity (ie big-O).

```
public int howManySteps(int n) {
    int m = 0;
    for (int i = n; i > 0; i/=2)
        for (int j = 0; j < n ; j++)
            m ++;
    for (int i = n; i > n/2; i--)
        m = m/2;
    return m;
}
```

The number of basic steps is $2+n+\lg n + 2 n \lg n$. The time complexity is $O(n \lg n)$.

Problem 3. (10 points) Assume that you have a Queue class containing the following method. What would be accomplished by the command `unknownMethod(0)`? Do not translate the method line by line. Instead explain its purpose and show the result of the method on a small example.

```
public void unknownMethod(int n){
    if (isEmpty())
        return;
    OurThing ot;
    if ( n % 2 == 0 ) {
        ot = dequeue();
        unknownMethod(n+1);
    }
    else {
        ot = dequeue();
        unknownMethod(n+1);
        enqueue(ot);
    }
}
```

It removes the head and every node after it and reverses the remaining items. If the queue contained 3 1 5 8 4 7 4 then the resulting queue would be 7 8 1 (should really show stack frames).

Problem 4. (10 points) Assume that the following method is in a class called MyStack. What is the purpose of the method? Do not simply translate the code line by line, instead give a high level description. Show the results of running the method on a small stack.

```
public int hmmm() {
    OurThing ot;
    int n;
    if (isEmpty())
        return 0;
    ot = pop();
    n = hmmm();
    push(ot);
    return n+1;
}
```

This method returns the number of items in the stack. If the stack contained 3 1 5 8 4 7 4 then the resulting stack would be 3 1 5 8 4 7 4 (should really show stack frames) and the value 7 would be returned.

Problem 5. (10 points) Assume that you have an empty hash table of size 11, a hash function $h(k) = k \% 11$, and a collision resolution scheme of double hashing with $d(k) = (k+1)^2 \% 11$. Show the hash table after each delete operation in the following sequence of operations: insert 48, insert 33, insert 23, insert 27, delete 48, insert 5, insert 1, delete 33, insert 9, delete 23.

33	23			*	27					
*	23			*	27			5	1	
*	*			*	27			5	9	

Problem 6. (10 points) Teri and Karen are both trying to implement a PriorityQueue class. Teri plans to use a singly linked list with a head reference. Karen plans to use a heap (using a vector). Under what circumstances should you prefer Teri's implementation? Under what circumstances should you prefer Karen's implementation? Explain.

Speed: Teri's method has an $O(n)$ enqueue and an $O(1)$ dequeue, while Karen's has $O(\lg n)$ enqueue and dequeue (except when resizing the vector).

Space: Teri's uses only a little extra space (for the next links), while Karen's (using a vector) has whatever extra space the vector has.

Ease: Teri's requires understanding references, while Karen's is a little more complicated to understand.

So Teri's is probably preferred when coding ease is the priority, but Karen's is preferred when performance is the priority.

Problem 7. (10 points) Assume that you have a Queue class that contains the following method. What is accomplished by the following method? Do not simply translate the method line by line. Instead explain its purpose and show the result on a small example. Under what circumstances will it throw an exception.

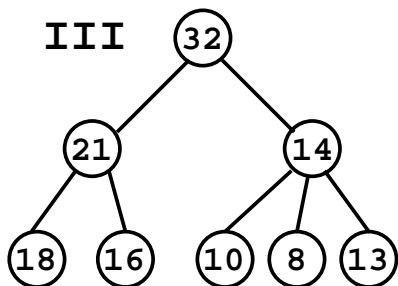
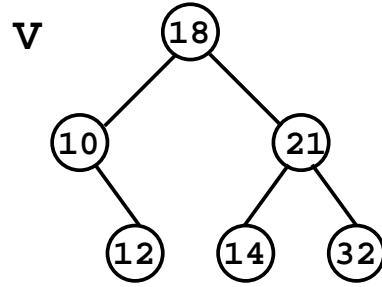
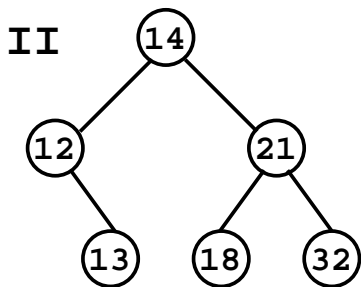
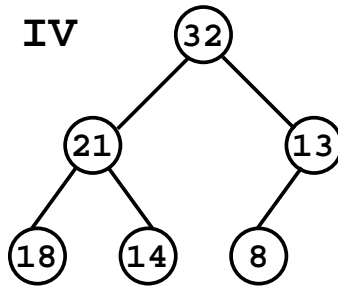
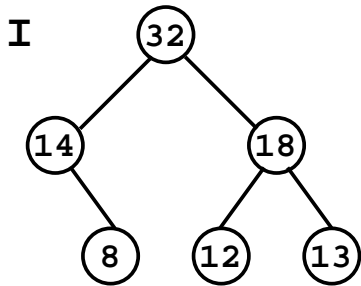
```
public void whatDoesThisDo() {
    OurThing ot1, ot2;
    if ( isEmpty() )
        return;
    ot1 = dequeue();
    ot2 = dequeue();
    whatDoesThisDo();
    if (ot1.getValue() < ot2.getValue())
        enqueue(ot2);
    else
        enqueue(ot1);
}
```

This method pairs the items (first two, next two, next two, ...) removes the smaller from each pair and reverses the remaining items. It will throw an exception if the number of items is odd. If the queue is initially 4 8 7 9 1 3 6 2 then the result is 6 3 9 8.

Problem 8. (10 points) Let $A = \{5, 3, 6, 7, 2\}$, $B = \{9, 7, 8, 4, 5\}$, and $C = \{9, 7, 2, 8, 1\}$. Find the following 5 sets.

$$\begin{aligned}
 |A| &= 5 \\
 A \cup C &= \{1, 2, 3, 5, 6, 7, 8, 9\} \\
 B \cap C &= \{7, 8, 9\} \\
 (A \cup B) \cap C &= \{2, 7, 8, 9\} \\
 (A - B) \cup C &= \{1, 2, 3, 6, 7, 8, 9\}
 \end{aligned}$$

Problem 9. (10 points) For each of the 5 trees below, indicate whether it is a Binary Search Tree, a Heap, both, or neither.



I _____
II _____
III _____
IV _____
V _____

Neither, BST, Neither, Heap, Neither

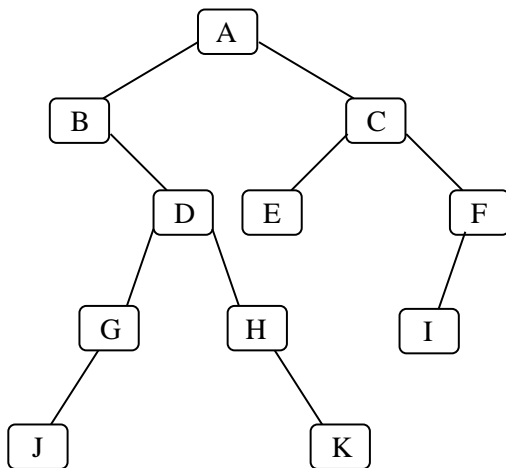
Problem 10. (10 points) Assume you have a BinarySearchTree class implemented using Nodes. Write a method for this class which receives a node and determines whether the binary search tree rooted at that node contains an even value.

```
public boolean hasEven(Node nd) {
    if (nd == null) return false;
    if (nd.getValue() % 2 == 0) return true;
    return (hasEven(nd.getLeft()) || hasEven(nd.getRight()));
}
```

Problem 11. (10 points) Assume that you have a class called BinarySearchTree implemented using Nodes [with the usual methods: getLeft(), getRight(), getValue()]. Write a method for this class which, when passed a Node, returns the sum of all values in the tree rooted at this Node.

```
public int sumOfValues(Node nd) {
    if (nd == null) return 0;
    return nd.getValue()+sumOfValues(nd.getRight()+sumOfValues(nd.getLeft());
}
```

Problem 12. (10 points) Give the inorder, preorder, and postorder traversals of the tree below.



Inorder =

Preorder =

Postorder =

Inorder = B J G D H K A E C I F
 Preorder = A B D G J H K C E F I
 Postorder = J G K H D B E I F C A

Problem 13. (10 points) Assume that you want to sort a set of items. Explain how selection sort and tree sort work. Under what circumstances would you prefer to use selection sort? Under what circumstances would you prefer to use tree sort? Explain.

Selection sort works by repeatedly swapping the largest item into the correct position. Tree sort works by inserting all items into a BST and then performing an inorder traversal. Selection sort is simpler to write and understand, but has time complexity $O(n^2)$ (worst and typical). Tree sort is more complicated, but typical performance is $O(n \lg n)$ (still $O(n^2)$ in the worst case).

Problem 14. (10 points) Sort the following array using heap sort. Show your intermediate steps.

21 | 14 | 7 | 10 | 8 | 9 | 3 | 32

After inserting the first 4 items. 21 | 14 | 7 | 10 | | | |

After inserting all 8 items 32 | 21 | 9 | 14 | 8 | 7 | 3 | 10

After deleting max 3 times 10 | 8 | 9 | 3 | 7 | | |

After deleting max 6 times 7 | 3 | | | | |

Sorted 3 | 7 | 8 | 9 | 10 | 14 | 21 | 32