Review Questions for Exam 3

CSCI 162

Dr. Schwartz

Interfaces

You may assume for this problem that the instance variables of a *Bag* class implemented with a linked list are *head* and *manyNodes*. *head* is a reference to *Node<E>* with nodes having data components of type *E*. *manyNodes* is an integer indicating the number of items in the bag.

Assume that the LinkedBag class implements the Iterable interface – that means it has a method with the interface
public Iterator<E> iterator()

a. Write a code segment that obtains an Iterator from a LinkedBag<Integer> object called *items* (this object already exists), and then iterates through the container displaying every element that is greater than or equal to 10.

   Remember the three methods in the interface for Iterator before writing this code (you will get some points for at least identifying these)!

```
Iterator<Integer> it = items.iterator();
while (it.hasNext()) {
      Integer num = it.next();
      if (num >= 10)
            System.out.print(num + " ");
}
```

Stacks and Queues

a. If you are writing an implementation for a stack and you choose to use a linked list, would you need a tail reference? Why or why not?

   You wouldn’t need a tail reference because a stack is LIFO (last in, first out). So all insertions and removals can occur at the head.

b. Use a stack to evaluate the following postfix expression. Show your work so that I can see you are using a stack. You will not receive full credit without appropriate stack pictures. You may not use calculators.

   `7 3 + 4 5 * 4 2 + - 3 * 6 / +`

```
                           2
               5       4   4    6         3          6
   3       4   4  20  20  20   20   14   14     42   42    7
7  7  10  10  10  10  10  10   10   10   10     10   10   10   17

_  _  __  __  __  __  __  __   __   __   ___    __   __   __   __
```

Intellectual property.

a. What do you need to do to copyright material you put on your web pages?
Nothing – if you have created it and publish it on your website, you own the copyright

b. Give two requirements for obtaining a patent on a software application.
Non-obviousness
utility
originality

Recursion
For this question, you will write two versions of a function: one iterative and one recursive. The functions will take in two integer parameters named *little* and *big*. The function prototype for **both** functions is:
void pattern(int little, int big)
When *little* is 1 and *big* is 5, the pattern is **5 4 3 2 1 2 3 4 5** with a space between each value and no newline.
When *little* is 17 and *big* is 20, the pattern is **20 19 18 17 18 19 20**

a. Write an iterative version of the function using a loop(s). Think simple and straightforward. This is not a trick.

```
public static void pattern(int little, int big) {
   for (int i = big; i >= little; i--)
     System.out.print(i + “ “);
   for (int j = little + 1; j <= big; j++)
     System.out.print(j + “ “);
}
```

b. Write a recursive version of the same function. It should use NO (absolutely no) loops. What is the base case?

```
public static void pattern(int little, int big) {
   if (little > big)
     return;
   else if (little == big) {
     System.out.print(little + “ “);
     return;
   }
   else {
    System.out.print(big + “ “);
    pattern(little, big-1);
    System.out.print(big + “ “);
   }
}
```

(1) What is the output of the following code if it is called with mystery(4);?

```
public static void mystery(int n) {
      if (n > 0) {
            System.out.print(n + " ");
            mystery(n-3);
            mystery(n-2);
            System.out.print(n + " ");
      }
}
```

4 1 1 2 2 4

For the binary tree below neatly write your answers to the following:

```
          42
         /  \
       17    58
       / \     \
     12   29    72
               / \
             68   99
```

a. Is it a binary search tree? Why or why not?
Yes, at each node all values to left are <= node value, all values to right are >

b. Show the result of an inorder traversal of this tree as shown.
12 17 29 42 58 68 72 99

c. Show the result of an preorder traversal of this tree as shown.
42 17 12 29 58 72 68 99

d. Show the result of a postorder traversal of this tree as shown.
12 29 17 68 99 72 58 42

e. Write the recursive code for doing a preorder traversal of a binary tree that stores characters. You can assume that you are given the root node as `node`, as shown below:

```
   public  void printPreorder(CharBTNode node) {

         if (node == null)
               return;
         else {
               System.out.print(node.getData() + “ “);
               printPreOrder(node.getLeft());
               printPreOrder(node.getRight());
         }
   }
```

f. The text suggests that a binary search tree would be a more efficient way to represent a Bag than the various ways we studied in class especially when searching. Why?

Because the organization of a binary search tree allows you to focus the search, ruling out portions of the data with each step rather than examining each element until you find the target.

Question from review session – posting Binary Search code:

```
   public static int binSearch(int[] array, int low, int high, int value)
{
         // base cases: value doesn't exist or we find the value
         // make sure that low <= high
         if (low <= high) {
               // find the midpoint between low and high
               int mid = low + (high - low)/2;
               if (array[mid] == value)
                     return mid;
               else if (value < array[mid])
                     return binSearch(array, low, mid-1, value);
               else
                     return binSearch(array, mid+1, high, value);
         }

         return -1;

   }
```