CISC 3130 Data Structures

This exam consists of 5 questions. Please complete the exam and submit it as a plain text email with the subject "CISC 3130 Final Exam" to nzhou@brooklyn.cuny.edu by 4PM on Monday, May 24.

Question 1: Program comprehension

1.1

Describe the behavior of the function `f` defined below.

```
static <T> void f(T arr[]){
    int n = arr.length;
    LinkedList<T> s = new LinkedList<T>();
    int i;

    for (i = 0; i < n; i++)
        s.add(arr[i]);

    i = 0;
    while (!s.isEmpty()){
        arr[i] = s.poll();
        i++;
    }
}
```

Assume `arr` contains values `[1,4,3,2,5]`. What is the content of `arr` after the function call `f(arr)`?

1.2

Consider the following function `f`:

```
static <T> void f(ArrayList<T> v) {
    int i, n;
    n = v.size();
    for (i = 1; i < n; i++){
        v.set(i-1, v.get(i));
    }
    v.remove(n-1);
    System.out.println(v);
}
```

Assume `v` has values `[1,2,3,4,5]`. What is the content of `v` after the function call `f(v)`?

1.3

Describe the behavior of the function `f` defined below.

```
public static ArrayList<Integer> f(int[] a){
    TreeSet<Integer> ts = new TreeSet<>();
    for (int x: a){
        ts.add(x);
    }
    ArrayList<Integer> al = new ArrayList<>();
    for (Integer x: ts){
        al.add(x);
    }
    return al;
}
```

Suppose `a` is the array {`3,1,1,2,2,1`}. What is the return value of the function call `f(a)`?

Question 2:

Consider the function `createMatrix` defined below:

```
static int[][] createMatrix(int n){
    int[][] a = new int[n][n];
    fill(a, 0, 0, n, 1);
    return a;
}

static void fill(int[][] a, int r, int c, int s, int val){
    if (s == 1){
        a[r][c] = val;
    } else if (s == 2){
        a[r][c] = a[r][c+1] = a[r+1][c+1] = a[r+1][c] = val;
    } else {
        for (int i = 0; i < s; i++){
            a[r][c+i] = a[r+i][c+s-1] = a[r+s-1][c+i] = a[r+i][c] = val;
        }
        fill(a, r+1, c+1, s-2, val+1);
    }
}
```

2.1 Display the array returned by the function call `createMatrix(6)`.

2.2 Equivalently rewrite the function `fill` such that it uses iteration, rather than recursion, to fill in the entries.

Question 3:

The function `countOccurrences` takes a document represented as a string, and returns a map that indicates the number of times each word occurs in the document.

```
    static Map<String, Integer> countOccurrences(String doc){
```

Assume that words in a document are separated by spaces. Implement the function.

Question 4:

Consider the class `ListNode`:

```
class ListNode<E> {
    E data;
    ListNode<E> next;

    public ListNode(E data){
        this.element = data;
        next = null;
    }

    public ListNode(E data, ListNode<E> next){
        this.element = data;
        this.next = next;
    }
}
```

Write each of the following functions:

4.1 `static <E> boolean prefix(ListNode<E> preHead, ListNode<E> head)`

This function returns true if the list represented by `preHead` is a prefix of the list represented by `head`.

4.2 `static <E> ListNode<E> take(ListNode<E> head, int k)`

This function returns the prefix of the list represented by `head` that has `k` elements. If the list contains fewer than `k` elements, then the function returns `head`. For example, for the list `head = [12,4,5,7]` and `k` = 3, `take(head, k)` returns `[12,4,5]`.

Question 5:

Consider the class `TreeNode`:

```
class TreeNode<E> {
    E element;
    TreeNode<E> left;
    TreeNode<E> right;

    public TreeNode(E element){
        this.element = element;
    }
}
```

Write each of the the following functions:

5.1 `static <E> int occurs(TreeNode<E> root, E elem)`

This function returns the number of times `elem` occurs in the binary tree whose root is `root`.

5.2 `static <E> ArrayList<E> rightMostPath(TreeNode<E> root)`

This function returns a list of node values on the path from the root to the right-most leaf. For example, for the tree in Fig. 1, the function returns [9, 12, 10].

5.3 `static <E> boolean  maxHeap(TreeNode<E> root)`

This function returns true if the binary tree under `root` has the max-heap property (i.e., it is complete, and in every subtree the root value is the maximum).

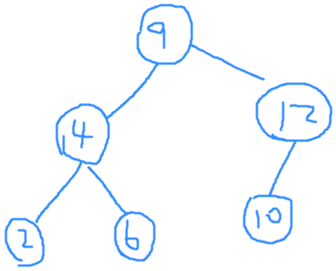

Figure 1: An example binary tree