Final Exam Summer 2020 Solutions

Q1. Linked List Length

Function `length()` calculates the length of a linked list. Fill in the blanks.

```
public class Node{
    public Node next;
    public int value;
}
public int length(Node r) {
    if(r == null) return 0;
    return length(r.next) + 1;
}
```

Q2. Print Linked List

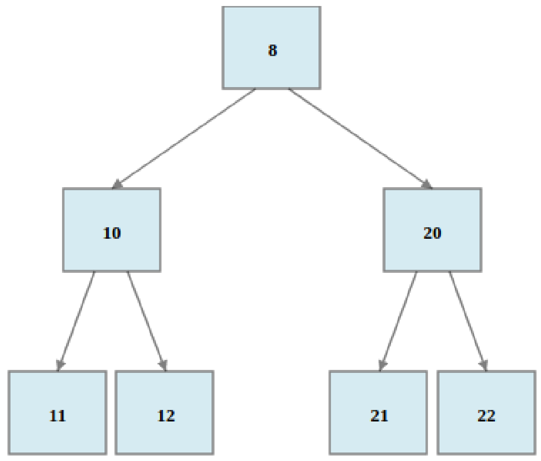
What does the following function do for a given Linked List with first node as head?

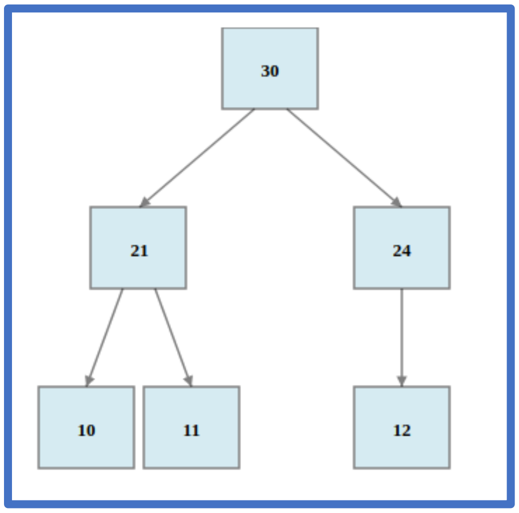
```
void foo(Node head) {
  if(head == null) return;
  foo(head.next);
  System.out.print(head.data);
}
```

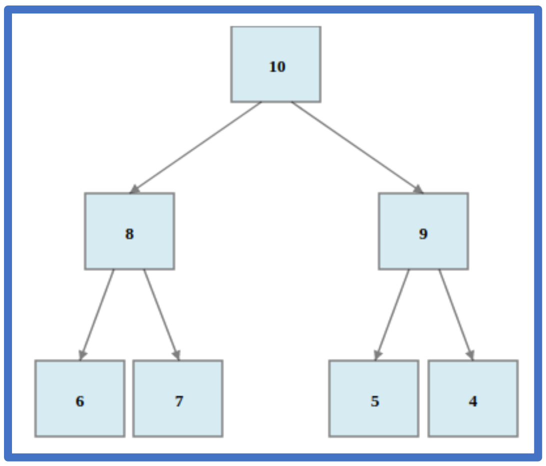
- Prints all nodes of linked list
- [✓ Prints all nodes of linked list in reverse order]
- Prints alternate nodes of linked list
- Prints alternate nodes in reverse order

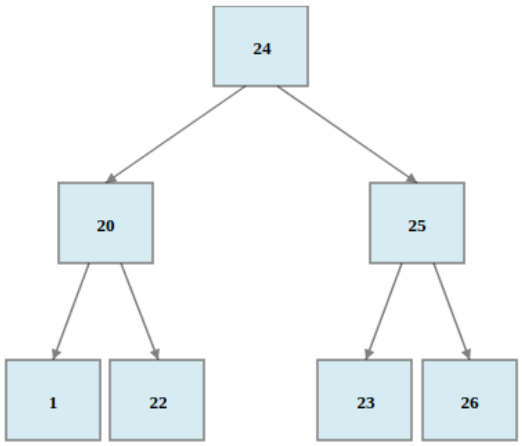
Q3. Heaps

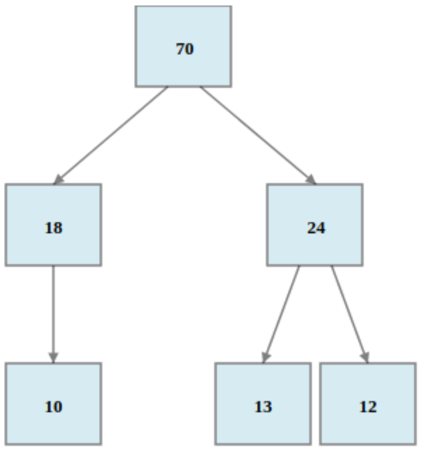
Which of the following trees are MAX heaps? (Choose all that apply)











Q4. Iterators

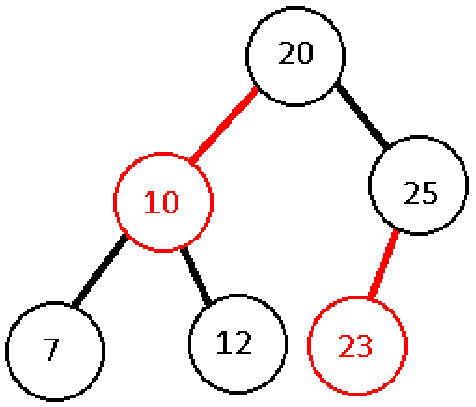
```
public class Node{
    public int value;
    Node next;
}
Node head;
...
// other code
...
public void mystery() {
    Iterator iter = new myIterator();
    while(iter.hasNext())
        System.out.print(iter.next() + “,”);
}

public class myIterator implements Iterator<Integer> {
    ArrayList<Integer> arr = new ArrayList<Integer>();
    int pos = 0;
    public myIterator() {
        Node data = head;
        while(data != null) {
            arr.add(0, data.value);
            data = data.next;
        }
    }
    public boolean hasNext() {
        return pos < arr.size();
    }
    public boolean next() {
        return arr.get(pos++);
    }
}
```

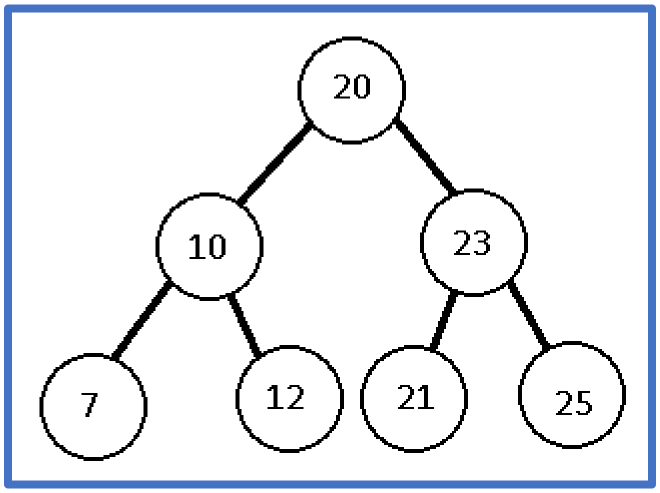
Assume `head` is the head of a list that looks like [1, 4, 2, 6, 7, 9, 0, 3, 7, 8].

What is the output when `mystery()` is run? 8, 7, 3, 0, 9, 7, 6, 2, 4, 1

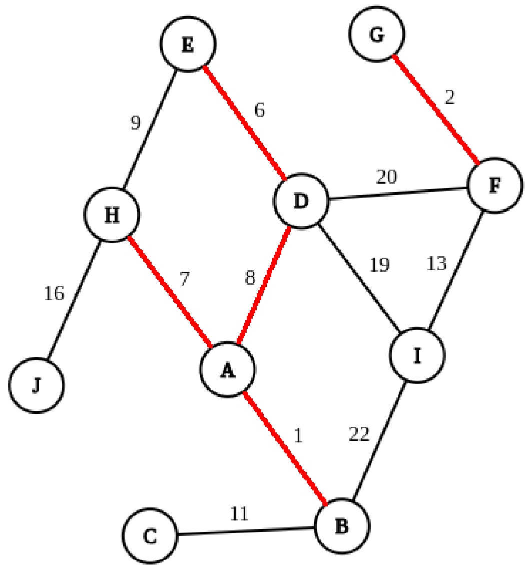
Q5. Red-Black Trees



What is the red-black tree that results from inserting 21 into the red-black tree above?



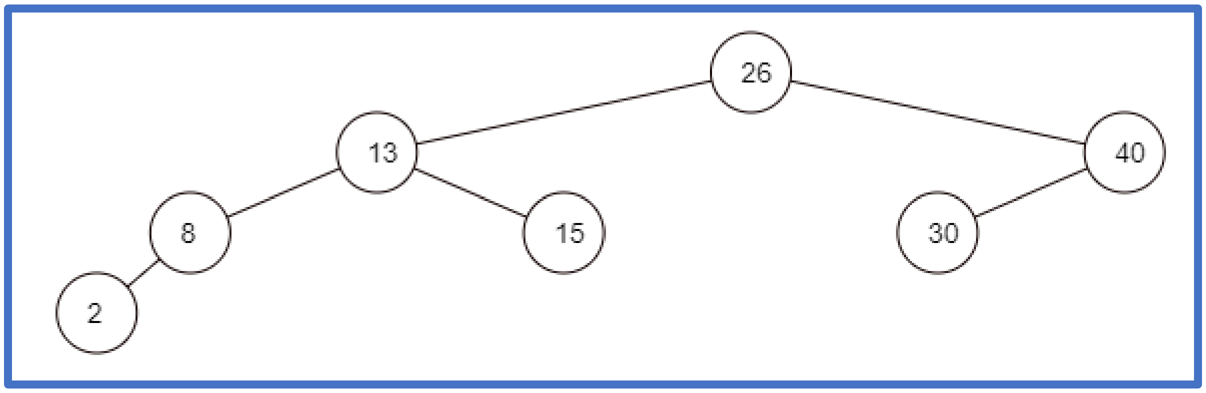
Q6. Minimum Spanning Tree



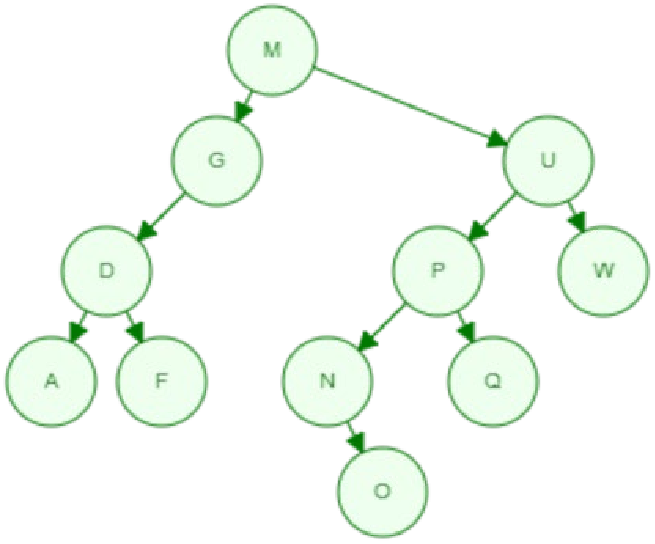
Assume we are applying Kruskal's algorithm to build a Minimum Spanning Tree from the graph above. We have picked the edges A-B, G-F, E-D, H-A, and D-A in that order. What edge will be picked next? B-C

Q7. Building a BST

Which of these BSTs is the correct result of adding: 26, 13, 8, 2, 40, 30, and 15 (in this specific order).



Q8. Tree Traversal



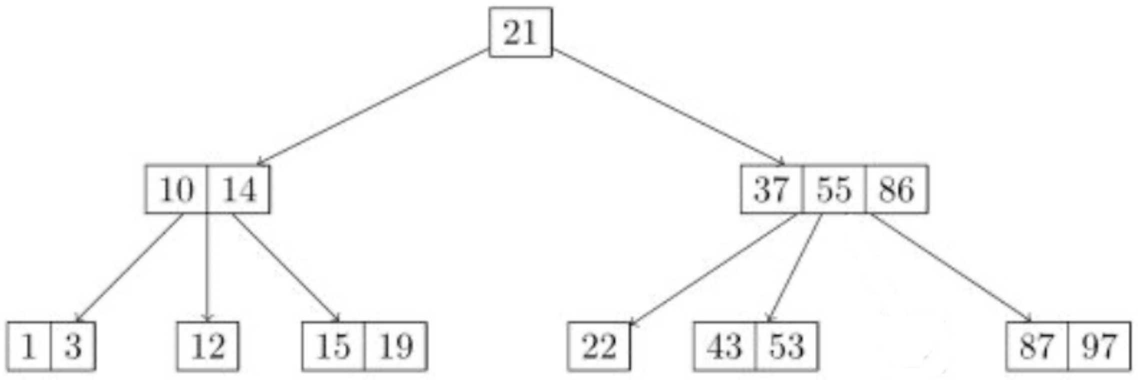
Given a tree, give the result of the preorder, inorder, and postorder traversal (comma delimited).

Preorder: M, G, D, A, F, U, P, N, O, Q, W

Inorder: A, D, F, G, M, N, O, P, Q, U, W

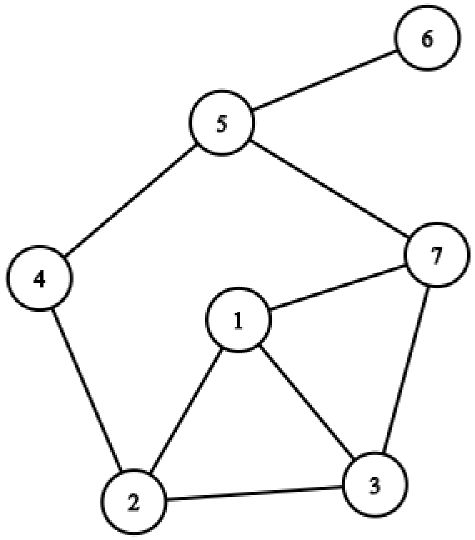
Postorder: A, F, D, G, O, N, Q, P, W, U, M

Q9. 2-3-4 Tree



The following 2-3-4 tree is not a valid 2-3-4 tree. Explain what is wrong with the tree? Right child of 55 is missing.

Q10. Graph Traversal



Starting from Vertex 2, return a BFS and DFS list. Prioritize by smaller vertex name. Answers should be formatted as an array. (Comma delimited, in square brackets []. Ex: [1,2,3])

BFS: [2, 1, 3, 4, 7, 5, 6]

DFS: [2, 1, 3, 7, 5, 4, 6]

Q11. Multi-Threading

Fill in the blank for this question. This program will create 2 threads, where each thread will just print out a string, and then merge the threads back to the main thread.

```
public class Customer implements Runnable {
    String name;
    int age;

    public Customer(String name, int age) {
        this.name = name;
        this.age = age;
    }

    //Method that we must implement
    public void run() {
        System.out.println("My name is " + this.name ". I am  " + this.age +
            "  years old.");
    }

    public static void main (String[] args) {
        //Create a thread t1 to encapsulate a customer named Subomi aged 30
        Thread t1 = new Thread(new Customer("Subomi", 30));
        //Create a thread t2 to encapsulate a customer named Cliff aged 60
        Thread t2 = new Thread(new Customer("Cliff", 60));
        //Make the two threads begin execution
        t1.start();
        t2.start();

        try {
            //Wait for the two threads to finish
            t1.join();
            t2.join();
        } catch(InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Q12. Linked List Removal

The function `everyOther()` transforms a linked list so that every other node is removed from the linked list. Start by removing the second element. So, 1 -> 2 -> 3 -> 4 -> 5 becomes 1 -> 3 -> 5. Fill in the blank. Lists of size 0 or 1, nothing changes.

```
public void everyOther(Node head) {
    if(head == null) {
        return;
    }
    if(head.next != null) {
        head.next = head.next.next;
    }
    everyOther(head.next);
}
```

Q13. Hashing

Given the following hash function and array, place the values in the order given into the array.

Values: `"Alice", "Carlos", "Al", "Frank", "Jackson", "John"`

```
public int hash(String str) {
  return (str.length() * 3) % 4;
}
```

If there is a collision, place it in the next available spot to the right, wrapping around when needed (linear probing). If a place is empty, use the empty string ("").

0	1	2	3	4	5	6	7

0: John

1: Jackson

2: Carlos

3: Alice

4: Al

5: Frank

6: “”

7: “”

Q14. Traversals

For the following class you need to implement `printPostOrder()`.

```
class Tree{
    private Class Node{
        String s;
        Node l, r;
    }
    private Node root;
    private void printPostOrder(Node curr){
        if (curr == null) return;
        printPostOrder(curr.l);
        printPostOrder(curr.r);
        System.out.print(curr.s);
    }
    public printTree(){
        printPostOrder(root);
    }
}
```

Q15. Merge

Complete the `merge()` method below. Given two integer arrays `a1` and `a2`, both already in ascending order, it should merge their contents into a new comparable array in ascending order. Assume neither array is null.

For example, if `a1 = [1, 5, 8, 10]`, `a2 = [2, 3, 5, 9, 20]`. Then `merge(a1, a2)` will return `[1, 2, 3, 5, 5, 8, 9, 10, 20]`

```
public int[] merge(int[] a1, int[] a2) {
    int len1 = a1.length;
    int len2 = a2.length;
    int[] a3 = new int[len1  + len2];
    int i = 0;
    int j = 0;
    int k = 0;

    while (i < len1 && j < len2) {
        if (a1[i] < a2[j]) {
            a3[k++] = a1[i++];
        } else {
            a3[k++] = a2[j++];
        }
    }

    while (i < len1) {
        a3[k++] = a1[i++];
    }

    while (j < len2) {
        a3[k++] = a2[j++];
    }
}
```

Q16. Convert

Given an integer BST, convert to a sorted Linked List.

```
public class BSTNode {
    public int value;
    BSTNode right;
    BSTNode left;
    public BSTNode(int v, BSTNode r, BSTNode l) {
        value = v;
        right = r;
        left = l;
    }
}
public class LLNode {
    public int value;
    LLNode next;
    public LLNode(int v, LLNode n) {
        value = v;
        next = n;
    }
}

//Given the root of a BST, create a sorted linked list of the values.
//Sorted means smallest value as the head and the largest value at the end.
//return the head of the linked list.
//You can use helper methods. Assume that bn is not null.
public LLNode convert(BSTNode bn) {
    LLNode head = new LLNode(0, null); // Dummy node
    LLNode curr = head;
    curr = convert(bn, curr);
    head = head.next;
    return head;
}
private LLNode convert(BSTNode bn, LLNode curr) {
    if (bn == null) return curr;
    curr = convert(bn.left, curr);
    curr.next = new LLNode(bn.value, null);
    curr = curr.next;
    convert(bn.right, curr);
}
```

Q17. Big O

```
void foo(int n){
  int k;
  for(k=1; k <= n; k=k*2){
     print("yes");
  }
}
```

What is the Big O of the following program? O(n)