New Mexico State University

C S 272/463 Introduction to Data Structures

Final Exam

Monday, December 6, 10:30am-12:30pm

Name:	
NMSU Email Address:	

This examination is closed book and notes. The examination duration is 2 hours. All students must answer all the questions. It contains 12 pages (including this one). The total exam value is 100 points. **No collaboration allowed on any exam in this course.**

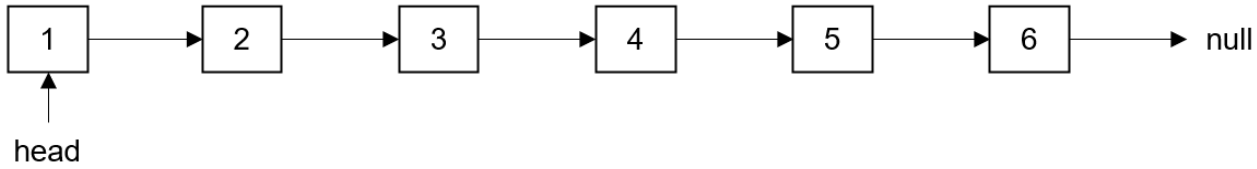
Before you start, please check your copy to make sure it is complete. Turn in all pages, together, when you are finished. **Write your initials on the top of ALL pages**.

Question	Max	Score
1	20	
2	20	
3	10	
4	20	
5	10	
6	10	
7	10	
Total	100	

Question 1 (20 pts) Given the *SNode* class as follows:

```
public class SNode <E>{
    public E data;
    public SNode<E> next = null;
    public SNode() {; }
    public static void f1(SNode head)
    {
        if (head == null)
        {
            return;
        }
        System.out.print(head.data + " ");

        if (head.next != null)
        {
            f1(head.next.next);
        }
        System.out.print(head.data + " ");
    }

    public void f2()
    {
        Node slow_ptr = this;
        Node fast_ptr = this;

            while (fast_ptr != null && fast_ptr.next != null)
            {
                fast_ptr = fast_ptr.next.next;
                slow_ptr = slow_ptr.next;
            }
            System.out.println(slow_ptr.data);

    }
}
```

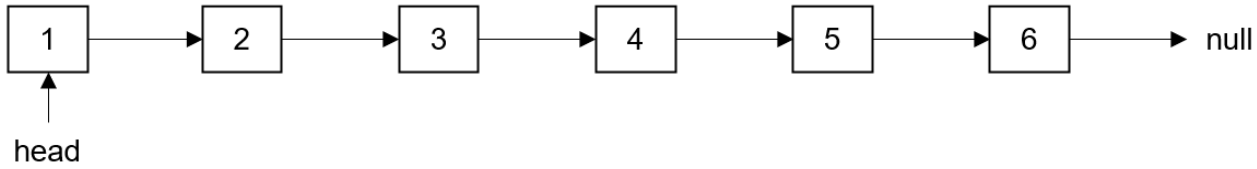
1A. (10 pts) What will **f1** print out when running SNode.**f1**(head) on the following list?



Answer:

1 3 5 5 3 1

1B. (10 points) What will **f2** print out when running head.f2() on the following list?



Answer:

4

Question 2 (20 pts)

2A. (10 pts) What does the following code fragment print when *n* is 30? In general, what does it do when presented with a positive integer *n*?

```
Stack s = new Stack();
while (n > 0) {
   s.push(n % 2);
   n = n / 2;
}
while (!s.isEmpty())
   System.out.print(s.pop());
```

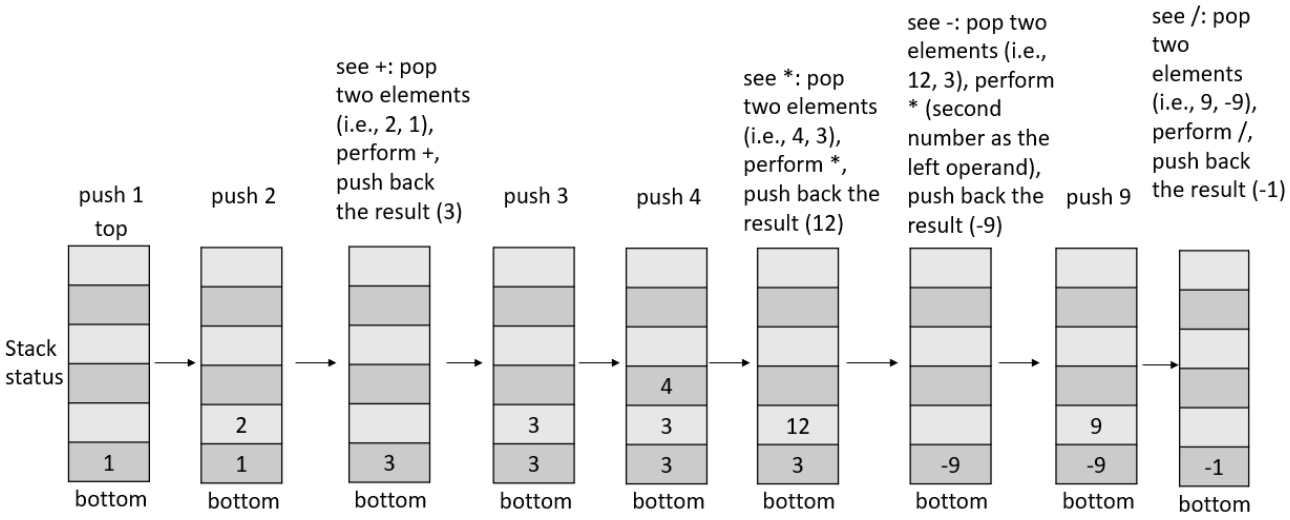
Answer:

11110

The code fragment converts a decimal number to a binary number.

2B. (10 pts) Show how to use a stack to evaluate the postfix expression 1 2 + 3 4 * - 9 /. Write the state of the stack after each token (i.e., an operator or operand) is processed.

Answer:



Question 3 (10 points)

What does the following code fragment print when *n* is 10?

```
Queue q = new Queue();
q.enqueue(0);
q.enqueue(1);
for (int i = 0; i < n; i++) {
   int a = q.dequeue();
   System.out.println(a);
   int b = q.front();
   q.enqueue(a + b);
}
```

Answer:

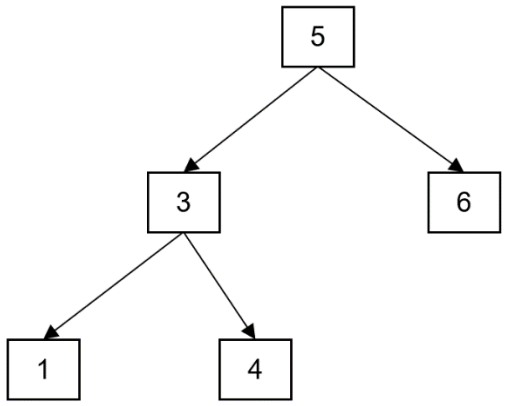
```
0

1

1

2

3

5

8

13

21

34
```

This program prints the first *n* Fibonacci numbers.

Question 4 (20 pts) Given the classes Node and BinaryTree as follows. Assume duplication values are not allowed in the tree.

```
class Node{
    public int data; //the element value for this node
    public BinaryTree left; //the left child of this node
    public BinaryTree right; //the right child of this node
    public Node () {
        data = 0; left = new BinaryTree(); right = new BinaryTree();
    }
    public Node (int initData) {
        data = initData;
        left = new BinaryTree(); right = new BinaryTree();
    }
}
public class BinaryTree {
    public Node root; // the root of the BST tree
    public BinaryTree() {root = null;}
    public boolean isEmpty() {return (root==null);}


    public boolean f3()
    {
        if (root == null)
            return true;

        if (root.left.root != null && root.left.root.data > root.data)
            return false;

        if (root.right.root != null && root.right.root.data < root.data)
            return false;

        if (!root.left.f3() || !root.right.f3())
            return false;

        return true;
    }
}
```

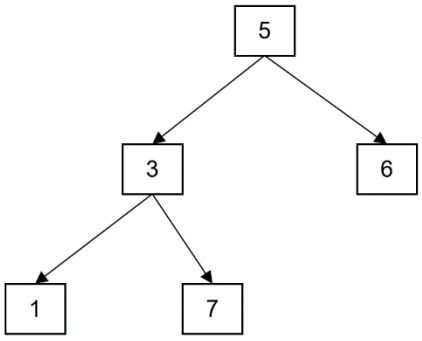
4A. (5 pts) Given the following BinaryTree tree **t1**, which is rooted at node with value 5. What will be returned when calling **t1.f3()?**



Answer:

True

4B. (5 pts) Given the following BinaryTree tree **t2**, which is rooted at node with value 5. What will be returned when calling **t2.f3()**?



Answer:

True

4C. (10 pts) In general, what does **f3** do?

Answer:

For every parent, check whether that parent is greater than its left child and smaller than its right child.

Question 5 (10 pts) Given the below *binarySearch* function

```
// Search e from array A[idxs,…., idxe]
// If e exists in A[idxs,…., idxe], return its index in A; otherwise,
// return -1

public int binarySearch (int[]A, int idxs, int idxe, int e){
    if(idxe<idxs) return (-1);
    int idx_middle = (idxe+idxs)/2;
    if(A[idx_middle]==e) return idx_middle;
    else if(e<A[idx_middle]) return binarySearch(A, idxs, idx_middle,e);
    else return binarySearch(A, idx_middle,idxe,e);
}
```

Given an array A with content {11, 15, 16, 18, 20, 25}.

Draw the recursion trace of **binarySearch(A, 0, 5, 18)**.

Answer:

Call binarySearch(A, 0, 5, 18) return 3;

Call binarySearch(A, 2, 5, 18) return 3;

Question 6 (10 pts) Given the following Heap class which utilizes an array to hold the elements. This heap needs to be a max heap.

```
public class Heap {
    private int[] elements;
    private int num;
    public Heap() {elements=new int[100]}; num=0;}
    public void add(int e){
        elements[num++] = e;
        reheapUpward(elements, num-1);
    }
    public static void reheapUpward(int[] elements, int pos){
        if(pos<=0) return;
        int parentPos = (pos-1)/2;
        if(elements[parentPos]<elements[pos]){
            int tmp = elements[parentPos];
            elements[parentPos]= elements[pos];
            elements[pos] = tmp;
            reheapUpward(elements, parentPos);
        }
    }
}
```

Sequentially add the following integers (in the order from left to right) to the heap [27, 35, 42, 21, 22, 23, 4]. Write the state of the array `elements` after each number is added to the heap.

Answer:

[27, …]

[27, 35, …] —reheapupward→ [35, 27, …]

[35, 27, 42, …] —reheapupward→ [42, 27, 35, …]

[42, 27, 35, 21, …]

[42, 27, 35, 21, 22, …]

[42, 27, 35, 21, 22, 23, …]

[42, 27, 35, 21, 22, 23, 4, …]

Question 7 (10 pts) Given the following Table class.

```
public class Table {
    private int num = 0;
    private Object[] keys = new Object[10];
    private Object[] data = new Object[10];
    private boolean[] used = new boolean[10];
    public Table(){
        for(int i=0;i<10;i++){used[i]=false; keys[i]=data[i]=null;}
    }
    private int hash(Object key){
        return Math.abs(key.hashCode())%data.length;}
    public void f4(Object _key, Object obj) throws Exception{
        if(num==data.length) throw new Exception("Table is full");

        int idx = hash(key);
        int count = 0;
        boolean found = false;
        while(count<data.length & used[idx]){
            if(key.equals(keys[idx])) {found = true; break;}
            else idx = ((idx+1)==data.length)?0:(idx+1);
            count ++;
        }
        if(found==false) idx = -1;

        if(idx!=-1) data[idx] = obj;
        else{
            idx = hash(key);
            while(used[idx]) {idx = ((idx+1)==data.length)?0:(idx+1);}
            keys[idx] = key;
            data[idx] = obj;
            used[idx] = true;
            num++;
        }
    }
}
```

Assume that your run the following several lines of code:

Table tb = new Table();
tb.f4(2, "o2");
tb.f4(15, "o15");
tb.f4(5, "o5");
tb.f4(7, "o7");
tb.f4(25, "o25");

What will be the content of keys, data, and used (Note that you also need to show clearly where the null is)?

Keys[0-9]: ___null, null, 2, null, null, 15, 5, 7, 25, null_____

Data[0-9]: ___ null, null, o2, null, null, o15, o5, o7, o25, null _____

Used[0-9]: ___ false, false, true, false, false, true, true, true, true, false _____

Answer:

---END--