

## Questions

1. Suppose you wish to “count down” the numbers from a given  $n$  down to 1. You can use a `while` loop to do this:

```
countdown(n){
    while (n > 0) {
        print n
        n-- }
}
```

Write a recursive version of this algorithm.

2. In class, we saw a non-recursive algorithm for computing  $x^n$  which ran in time proportional to  $n$ , and we saw a recursive algorithm that ran in time proportional to  $\log n$ . Now write a recursive algorithm that runs in time proportional to  $n$ .
3. Write a recursive algorithm that prints out a consecutive sequence of elements in an array. The method has three parameters: the array name, and the first and last indices to print from i.e. `displayArray( a, first, last)`
4. Write a recursive algorithm `reverseRecursive` which reverses `next` links in a singly linked list. Such an algorithm may be used as a helper method for the following, which reverses the order of nodes in a list and also reverses the head and tail.

```
void reverse(){
    if (head != null){
        reverseRecursive(head);
        head.next = null;
        SNode tmp = tail;
        tail = head;
        head = tmp;
    }
}
```

5. In lecture 11, I discussed the game “20 questions” and we looked at an example of finding a number from 0 to  $n = 2^m - 1$  using  $m$  questions. We saw that you required  $m = \log n$  questions to do so.

Here is a slightly different game. Suppose I am thinking of a positive number  $n$  but I don't give you a bound on the number. It is easy for you figure out the number using  $n$  questions, namely question  $i$  is “is it the number  $i$ ?”. Give a faster algorithm, namely that runs in time proportional to  $\log n$ .

6. Google “All horses are the same color”. This is a nice example of the subtleties in proofs by induction.

## Answers

1. 

```
countdown(n){
    if (n > 0) {
        print n
        countdown(n - 1)    }
    }
```
2. 

```
power(x, n){    //    assume n >= 0
    if (n == 0)
        return 1.0
    else
        return x * power(x, n - 1);
    }
```
3. 

```
displayArray( a, first, last){
    print a[first])
    if (first < last)
        displayArray(a, first+1, last)
    }
```

Alternatively, you could do it like this:

```
displayArray( a, first, last){
    if (first < last)
        displayArray(a, first, last-1);
    print a[last])
}
```

4. **MODIFIED ON FEB. 21. UNNECESSARY TEST FOR head != null removed.**

```
SNode reverseRecursive(head){    // only gets called when head != null
    SNode tmp = head;
    head = head.next;
    if (head != null){
        reverseRecursive(head); // see above
        head.next = tmp;
    }
    return tmp;
}
```

```
5.  i = 0
    answer = 0
    while (answer = 0){
        i++
        ask "is n < power(2,i) ?"
        answer = the answer to the above question (0 no, 1 yes)
    }
    // Here answer == yes
    Do a binary search for n, given this upper bound, n <= power(2,i)
```

Note that the while loop takes time  $c \log n$  and so does binary search.

6. Wikipedia gives a good discussion which I won't repeat here.