Section 8.10

8.10.1

a. *Give a recursive algorithm to compute the sum of the cubes of the first n positive integers. The input to the algorithm is a positive integer n. The output is $\sum_{j=1}^{n}$. The algorithm should be recursive, it should not compute the sum using a closed form expression or an iterative loop.*

```
CubeSum(n)
{
   if(n == 1) return(1);
   return(n**3 + CubeSum(n-1));
}
```

8.10.2

a. *Give a recursive algorithm which takes as input a positive integer n and returns the sum of the first n positive odd integers.*

```
OddSum(n)
{
   if(n == 1) return(1);
   return(2*n+1 + OddSum(n-1));
}
```

8.10.3

The input to the maximum and minimum problems is a sequence of numbers, $a_1 \ldots a_n$, where n, the length of the sequence, is a positive integer. The function length($a_1 \ldots a_n$) returns n, the length of the sequence. If $n > 1$, you can also create a new sequence $a_1 \ldots a_{n-1}$, which is the original sequence $a_1 \ldots a_n$, with the last number a_n omitted.

a. *Give a recursive algorithm which takes as input a sequence of numbers and returns the minimum (i.e., smallest) number in the sequence. Your algorithm should not use an iterative loop.*

```
SequenceMin(a, n)
{
   if(n == 1) return a_1;
   b = a_1 ... a_{n-1};
   return Min(a_n, SequenceMax(b, n-1));
}
```

b. *Give a recursive algorithm which takes as input a sequence of numbers and returns the maximum (i.e., largest) number in the sequence. Your algorithm should not use an iterative loop.*

```
SequenceMax(a, n)
{
   if(n == 1) return a_1;
   b = a_1 ... a_{n-1};
   return Max(a_n, SequenceMax(b, n-1));
}
```