Problem 1:

(1)

```
method1(int[ ] a)  // returns integer
	x = 0;
	y = 0;
	for (i=1; i<n; i++) { // n is the number of elements in array a
		if (a[i] == a[i-1]) {
			x = x + 1;
		}
		else {
			y = y + 1;
		}
	}
	return (x - y);
```

O(n) linear,
One for loop so f(n) = n+1

(2)

```
method2(int[ ] a, int[ ] b)  // assume equal-length arrays
	x = 0;
	i = 0;
	while (i < n) { // n is the number of elements in each array
		y = 0;
		j = 0;
		while (j < n) {
			k = 0
			while (k <= j) {
				y = y + a[k];
				k = k + 1;
			}
			j = j + 1;
		}
		if (b[i] == y) {
			x++;
		}
		i = i + 1;
	}
	return x;
```

O(n^3) cubic,
Three while loop that add one at a time so its n^3

(3)

```
// n is the length of array a
// p is an array of integers of length 2
// initial call: method3(a, n-1, p)
// initially p[0] = 0, p[1] = 0
method3(int[] a, int i, int[] p)
```

```
if (i == 0) {
      p[0] = a[0];
      p[1] = a[0];
}
else {
      method3(a, i-1, p);
      if (a[i] < p[0]]) {
            p[0] = a[i];
      }
      if (a[i] > p[1]]) {
            p[1] = a[i];
      }

}
```

O(n) linear,
Recursion method3, i-1 one at a time so its linear

(4)

```
// initial call: method4(a, 0, n-1) // n is the length of array a
public static int method4(int[] a, int x, int y)
   if (x >= y) {return a[x];}
   else {
         z = (x + y) / 2; // integer division
         u = method4(a, x, z);
         v = method4(a, z+1, y);
         if (u < v) return u;
         else return v;
     }
   }
```

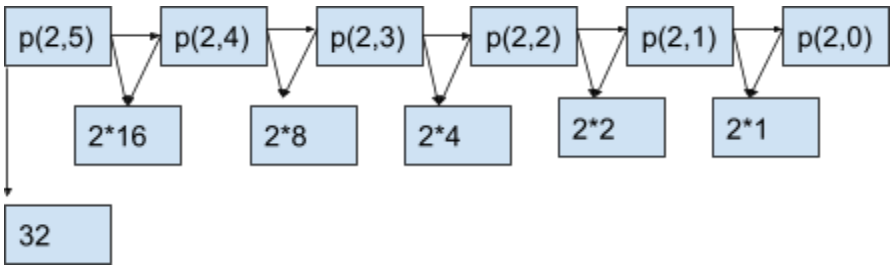
O(n) linear,
Am I seems like a very confusing Loop but it's actually a linear function because that inside the method4 its will call method 4 two times, so its n*2 but it is still linear

Problem 2:

(1)

```
Public static double power(double x, int n){
        if (n==0)
                Return 1;
        else
                Return x * power(x, n-1)
}
```

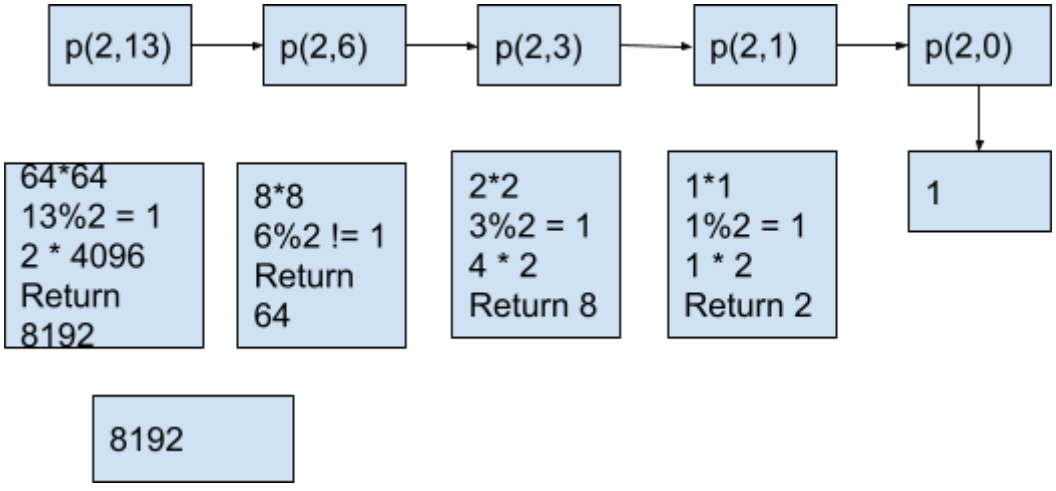
power(2, 5)



(2)

```
Public static double power(double x, int n){
        if (n==0)
                Return 1;
        else
                Double partial = power(x, n/2);
                Double result = partial * partial;
                If (n%2==1)
                        Result *=  x;
                Return result
}
```

power(2, 13)



Problem 3:

(1)

operation	Return value	Stack contents
push(10)	-	(10)
pop()	10	()
push(12)	-	(12)
push(20)	-	(12,20)
size()	2	(12,20)
push(7)	-	(12,20,7)
pop()	7	(12,20)
top()	20	(12,20)
pop()	20	(12)
pop()	12	()
push(35)	-	(35)
isEmpty()	false	(35)

(2)

operation	Return value	Queue Contents (first ← Q ← last)
enqueue(7)	-	(7)
dequeue( )	7	()
enqueue(15)	-	(15)
enqueue(3)	-	(15,3)
first()	15	(15,3)
dequeue( )	15	(3)
dequeue( )	3	()

first()	null	()
enqueue(11)	-	(11)
dequeue( )	11	()
isEmpty()	true	()
enqueue(5)	-	(5)