1. (15 points) Determine the big-O running time with respect to n of each of the following methods:

```java
public static void p1(int n){
    for(int i=1;i<n;i=i*2)
        System.out.println(i);
}
```

```java
public static void p2(int n){
    for(int i=100;i<n;i++)
        System.out.println(i);
}
```

```java
public static void p3(int n){
    for(int i=0;i<n;i=i+n)
        System.out.println(i);
}
```

```java
public static void p4(int n){
    while (n>0)
        n=n-2;
}
```

```java
public static void p5(int n){
    for(int i=0;i<n;i++)
        for(int j=i;j<n;j++)
            System.out.println(i);
}
```

```java
public static void p6(int n){
    for(int i=0;i<n;i++)
        p2(n);
    p5(n);
}
```

```java
public static void p7(int n){
    p1(n);
    p2(n);
}
```

2. (25 points) For each of the following recursive methods, write a recurrence of the form \( T(n) = a T(f(n)) + g(n) \) to describe its running time. Then solve your recurrences using the iteration method or the Master theorem.

```java
public static void r1(int n){
    if(n>0){
        r1(n-1);
        System.out.print(n+" ");
        r1(n-1);
    }
}
```

```java
public static void r2(int n){
    if (n>0) {
        System.out.println(n);
        r2(n/2);
        r2(n/2);
    }
}
```
public static void r3(int n) {
    if (n > 0) {
        for (int i = 0; i < n; i++)
            System.out.print(i);
        r3(n / 2);
        r3(n / 2);
    }
}

public static void r4(int n) {
    if (n > 0) {
        for (int i = 0; i < 4; i++)
            r4(n / 2);
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                System.out.println(i);
    }
}

public static void r5(int n) {
    if (n > 0) {
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                System.out.println(i);
        r5(n / 2);
        r5(n / 2);
    }
}

3. (10 points) Selection sort takes $1 \mu s \ (1 \times 10^{-6} s)$ to sort an array of size 1000. What is the largest array that it can sort in $10 ms \ (10 \times 10^{-3} s)$?

4. (12 points) If a program solves the towers of Hanoi problem in for 30 disks in 1 minute, how long does it take to solve the problem with 24 disks? How about 60 disks?

5. Solve $T(n) = T(n-1) + n$ by iteration.

6. Solve $T(n) = 2T(n/2) + 1$ using a recursion tree.

7. We have four algorithms that run in $O(1)$, $O(n)$, $O(n^2)$ and $O(n^3)$. Suppose that for a particular computer and a value of $n = 1000$, it took the following times for each algorithm to finish:

   (a) $O(1)$ - 100 $\mu s$
   (b) $O(n)$ - 50 $\mu s$
   (c) $O(n^2)$ - 20 $\mu s$
   (d) $O(n^3)$ - 5 $\mu s$

   Approximately, how long would it take each of the algorithms to run if $n = 1,000,000$? Recall that $1 \mu s = 1 \times 10^{-6} s$.

8. (30 points) Write two recursive methods to add all the elements in an array A as follows:

   (a) Method 1: Return the last element of the (sub)array plus the sum of all remaining elements in the (sub)array. If the (sub)array has no elements, return 0. Hint: Your method should receive the array and the index of the last element to process, thus the initial call would be `arraySum(A, A.length-1)`.

   (b) Method 2: Return the sum of the first half of the (sub)array plus the sum of the second half of the (sub)array. If the (sub)array has no elements, return 0. Hint: Your method should receive the array and the index of the first and last element to process, thus the initial call would be `arraySum(A, 0, A.length-1)`.

   For each of your methods, write and solve recurrences describing their running times.
9. (15 points) Consider the following methods:

```java
public static void q1(int n){
    if(n>0){
        n++;
        q2(n/2);
        System.out.println(n);
    }
}

public static void q2(int n){
    System.out.println(n);
    q1(n/2);
    n++;
    System.out.println(n);
}
```

Trace the execution of q1(6). Draw the stack of activation records after each method call and show the output that would be produced.