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

```java
public static void p1(int [] A){
    for(int i=0; i<A.length; i++)
        for(int j=0; j<A.length; j++)
            System.out.println(A[i]);
    for(int p=0; p<A.length; p++)
        for(int m=1; m<A.length; m=m*2)
            System.out.println(A[m]);
}

public static void p2(int [] A){
    for(int i=0; i<A.length; i++)
        for(int j=0; j<10000; j++)
            System.out.println(A[i]);
    for(int p=0; p<A.length; p++)
        for(int m=1; m<A.length; m=m*2)
            System.out.println(A[m]);
}

public static void p3(int [] A){
    for(int i=0; i<10000; i++)
        for(int j=0; j<i; j++)
            System.out.println(A[j]);
}

public static void p4(int [] A){
    for(int i=0; i<A.length; i=i+20)
        System.out.println(A[i]);
}

public static void p5(int [] A){
    for(int i=0; i<A.length; i++)
        for(int j=0; j<A.length; j++)
            for(int k=A.length; k>0; k=k/2)
                System.out.println(A[i]);
}

public static void p6(int [] A){
    for(int i=1; i<A.length; i=i+i)
        System.out.println(A[i]);
}

public static void p7(int [] A){
    for(int i=0; i<A.length*A.length; i++)
        System.out.println(A[0]);
}

public static void p8(int [] A){
    for(int i=1; i<A.length*A.length; i=2*i)
        System.out.println(A[0]);
}
```
2. For each of the following recursive methods, write a recurrence of the form \( T(n) = aT(f(n)) + g(n) \) to describe its running time. Then solve your recurrences using the iteration method or the Master theorem.

```java
public static void r1(int[] A, int n){
    if(n>0){
        r1(A,n-1);
        System.out.print(A[n]);
        r1(A,n-1);
    }
}

public static void r2(int[] A, int n){
    if (n>0) {
        for(int i=0;i<2;i++)
            r2(A,n/2);
        for(int i=0;i<n;i++)
            System.out.print(A[i]);
    }
}

public static void r3(int[] A, int n){
    if (n>0) {
        System.out.print(A[0]);
        r3(A,n/2);
        r3(A,n/2);
    }
}

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

public static void r5(int[] A, int n){
    if(n>0){
        for(int i=0;i<8;i++)
            r5(A,n/2);
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                for(int k=0;j<n;j++)
                    System.out.println(A[i]);
    }
}

public static void r6(int[] A, int n){
    if (n>0) {
        System.out.print(A[0]);
        r6(A,n/2);
    }
}
```

3. 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 \( 10ms \ (10 \times 10^{-3}s) \)?

4. 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. If a program performs binary search for an array of size 10,000 in 1 ms, how long does it take to search an array of size 100? How about 100,000,000?
6. The following programs have the given running times for the specified values of \( n \). What are their likely big-O running times?

\[
\begin{array}{|c|c|c|}
\hline
n = 100 & n = 10,000 \\
\hline
A & 1 \text{ ms} & 1 \text{ ms} \\
B & 1 \mu s & 2 \mu s \\
C & 10 \text{ ms} & 1 \text{ s} \\
D & 1 \text{ ms} & 10 \text{ s} \\
E & 1 \mu s & 1 \text{ s} \\
\hline
\end{array}
\]

7. Solve \( T(n) = T(n-1) + n \) by iteration.

8. Solve \( T(n) = 2T(n/2) + 1 \) using a recursion tree.

9. 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 \( \mu \text{s} \)
(b) \( O(n) \) - 50 \( \mu \text{s} \)
(c) \( O(n^2) \) - 20 \( \mu \text{s} \)
(d) \( O(n^3) \) - 5 \( \mu \text{s} \)

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

10. 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 \( \text{arraySum}(A, A.length-1) \).

(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 \( \text{arraySum}(A, 0, A.length-1) \).

For each of your methods, write and solve recurrences describing their running times.

11. Consider the following methods:

\[
\begin{array}{l}
\text{public static void q1(int n)\{} \\
\hspace{1cm} \text{if(n>0)\{} \\
\hspace{2cm} n++; \\
\hspace{2cm} q2(n/2); \\
\hspace{2cm} \text{System.out.println(n);} \\
\hspace{1cm} \} \\
\text{\}} \\
\text{public static void q2(int n)\{} \\
\hspace{1cm} \text{System.out.println(n);} \\
\hspace{1cm} q1(n/2); \\
\hspace{1cm} n++; \\
\hspace{1cm} \text{System.out.println(n);} \\
\hspace{1cm} \} \\
\end{array}
\]

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

12. Write a recursive method that receives a reference to the first node of a list of iNode objects and an integer \( k \) and determines if \( k \) is in the list.

13. Write an iterative method that receives a reference to the first node of a list of iNode objects and an integer \( k \) and determines if \( k \) is in the list.

14. Write an iterative method that receives a reference to the first node of a list of iNode objects and determines if all the items in the list are identical.
15. What is the output of the following code fragment:

```java
iNode x = new iNode(10);
for(int i=3;i>0;i=i-1)
    x = new iNode(i,x);
iNode y = new iNode(20,x);
iNode z = new iNode(30,y);
printList(x);
printList(y);
printList(z);
y.next.item = 8;
printList(x);
y= y.next;
printList(x);
z= y.next;
z.next = null;
printList(x);
printList(y);
```