1. The code below shows a method for **binary search**. This version takes as input an integer `find` and an array of integers called `list` which are sorted in increasing order. If `find` is contained in the array, the method returns the index where it is stored in the array. If it is not contained in the array, it returns -1. Every time an integer in the array is examined, it is printed out.

```java
int binarySearch( int find, int[] list )
{
    int low = 0;
    int high = list.length()-1;
    int mid;

    while ( low < high )
    {
        mid = ( low + high )/2;
        System.out.println( list[mid] );

        if ( list[mid] == find )
            return( mid );

        if ( find < list[mid] )
            high = mid - 1;
        else
            low = mid + 1;
    }

    if ( low == high && list[low] == find )
        return( low );

    else
        return( -1 );
}
```
(a) What is the running time of the binary search method on the opposite page? Use \( \Theta \)-notation and assume that \( n \) is the number if integers in the array.

(b) Show the output of the method when it is called on the following input:

\[
\text{find} = 11
\]

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\text{list:} & -4 & -2 & -1 & 0 & 11 & 17 & 19 \\
\end{array}
\]

2. Consider the tree shown below. Suppose that the tree is traversed in \textbf{pre-order}. When a node is “visited”, its label is printed out. Show the order that the labels are printed out.

\[
\begin{array}{c}
A \\
G & E \\
K & J & H \\
D & B \\
\end{array}
\]