CMPT 202 - Searching

1. Why might we need to search for an item in an array/list?

2. Consider the following array. Write the psuedocode for an algorithm for a *sequential search* that returns true if a given value occurs in the list, and false otherwise.

115	67	0	-200	54	99	12	42	-42	15	60
0	1	2	3	4	5	6	7	8	9	10

```
boolean search(int[] array, int value) {







}
```

3. Complete the following table using the sequential search with N elements:

	Number of Comparisons
Best case	
Worst case	
Average case	

4. What is the Big-Oh notation of the sequential search?

We can use a different algorithm if our data is in sorted order:

-200	-42	0	12	15	42	54	60	67	99	115
0	1	2	3	4	5	6	7	8	9	10

5. What is the index and value at the midpoint of the list?

6. Suppose we are searching for 12, and we begin searching at the midpoint of the list. What can an algorithm take advantage of if the list is sorted?

This is known as a *binary search*, and the general recursive algorithm is as follows:

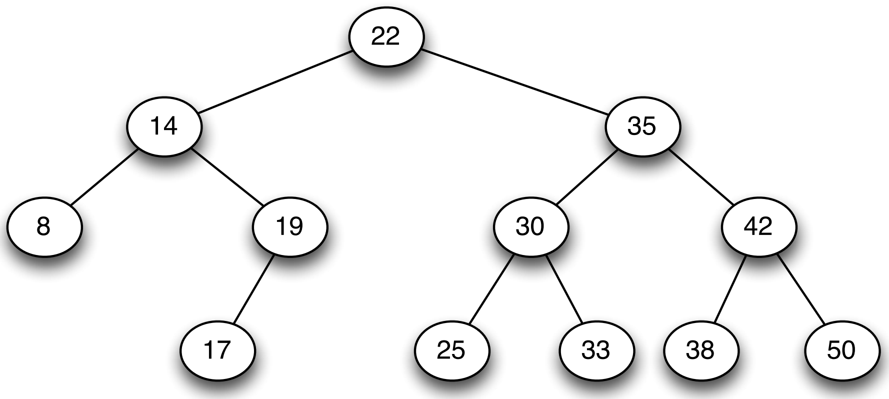
```
// search for desiredItem in elements
binarySearch(elements, desiredItem) {
   // calculate midpoint

   if (desiredItem == element at midpoint)
      we found it!!
   else if (desiredItem < element at midpoint)
      search left side of elements
   else
      search right side of elements
}
```

7. After proceeding through the worksheet, complete the following table using the binary search with the following size of N:

	$N = 4$	$N = 32$	$N = 64$	$N = 256$	$N = 1024$
Max number of comparisons					

8. What is the Big-Oh notation of the binary search?

Consider the following binary tree



9. Explain how binary search can be performed on a binary search tree.

10. What is the height of a complete binary search tree with N nodes?

11. Explain how the shape of a binary search tree can influence its Big-oh performance.