← Back Programming Assignment Quiz (Do programming assignment FIRST)
Graded Quiz • 20 min

**Due** Jun 20, 12:29 PM IST

## Congratulations! You passed!

**Grade received** 100% **To pass** 90% or higher

Go to next item

# Programming Assignment Quiz (Do programming assignment FIRST)

**Latest Submission Grade 100%**

**1.** Which of the following is a requirement for using the binary search algorithm to find an item in an array?

**1 / 1 point**

● The array must be in sorted order

○ The array must contain either ints or Strings

○ The array must have more than 2 elements in it

○ None of the above is a requirement of binary search

✓ **Correct**
This is the correct response. Binary search rules out half of the remaining portion of the list each time, but it can only do this on a sorted list.

**2.** The following code implements binary search. Choose the option that correctly fills in the missing line of code:

**1 / 1 point**

```
/* Return true if toFind is found in toSearch,
 * otherwise return false.  Uses binary search. */
public static boolean binarySeach(int[] toSearch, int toFind)
{
    int low = 0;
    int high = toSearch.length - 1;
    int mid;
    while ( << INSERT ANSWER CODE HERE >> )
    {
        mid = low + ((high-low)/2);
        if (toFind < toSearch[mid]) {
            high = mid-1;
        }
        else if (toFind > toSearch[mid]) {
            low = mid+1;
        }
        else return true;
    }
    return false;
}
```

●
```
high >= low
```

○
```
low < high
```

○
```
mid >= low
```

○
```
found == false
```

✓ **Correct**
This is the correct response. High must be greater than or equal to low. If it falls below low, then we conclude toFind is not there.

**3.** Consider the following ordering of integers in an array:

**1 / 1 point**

8 , 12 , 45 , 58 , 22 , 18 , 43 , 30

Which of the following sorting algorithms could have produced this array above after 3 iterations of its outer loop?

● Insertion Sort

○ Selection Sort

○ Both Insertion and Selection Sort

○ Neither Insertion nor Selection Sort

✓ **Correct**
Insertion sort could have produced this ordering because the first four elements in the list are in sorted order. At the end of the i'th iteration of the outer loop, insertion sort ensures that the first i+1 elements are in sorted order relative to each other.

**4.** True or false: Selection sort is faster when the elements in the array are already sorted than when they are unsorted.

**1 / 1 point**

○ True

● False

✓ **Correct**
This is false because the two nested loops in selection sort always execute to completion, no matter what the order of the underlying elements. Because the inner loop in selection sort must find the smallest remaining element, it must search all the way through the remaining elements to be sure its found the smallest.

**5.** True or false: Insertion sort as we examined in (i.e. mystery sort) is faster when the elements in the array are already sorted than when they are unsorted.

**1 / 1 point**

● True

○ False

✓ **Correct**
This is true because the inner loop in insertion sort aborts as soon as it discovers that the element it is moving is already in the correct relative position. If the array is already sorted, then the inner loop will always abort after just one check.

**6.** Which of the following correctly implements the compareTo method in EarthquakeMarker so that earthquakes are sorted from highest magnitude to lowest magnitude?

**1 / 1 point**

○ You don't need to implement compareTo in EarthquakeMarker. This is the default sorting behavior for EarthquakeMarkers.

○

```
public boolean compareTo(EarthquakeMarker m)
{
    if (this.getMagnitude() > m.getMagnitude())
        return true;
    else
        return false;
}
```

○

```
public int compareTo(EarthquakeMarker m1, EarthquakeMarker m2)
{
    if (m1.getMagnitude() > m2.getMagnitude())
        return 1;
    else if (m1.getMagnitude < m2.getMagnitude())
        return -1;
    else:
        return 0;
}
```

●

```
public int compareTo(EarthquakeMarker m)
{
    if (m.getMagnitude() < this.getMagnitude())
        return -1;
    else if (this.getMagnitude() < m.getMagnitude())
        return 1;
    else
        return 0;
}
```

○

```
public int compareTo(EarthquakeMarker m)
{
    if (this.getMagnitude() < m.getMagnitude())
        return -1;
    else if (this.getMagnitude() > m.getMagnitude())
        return 1;
    else
        return 0;
}
```

Correct

This code correctly implements the compareTo method. It will return 1 if the magnitude of calling object's earthquake is less than the argument object's magnitude, and -1 when the calling object's magnitude is greater. This will lead to smaller magnitudes being later in the list, because the calling object is considered "less than" the argument (i.e. return value of -1) when its magnitude is greater.

7. Run your program with the earthquake input file "quiz2.atom". There is a commented out line of code in the setUp method in EarthquakeCityMap that you can uncomment to make this happen:

1 / 1 point

```
earthquakesURL = "quiz2.atom";
```

Call sortAndPrint as appropriate to answer the following question.

What is the magnitude of the 6th earthquake printed (i.e. the 6th strongest earthquake)? To be clear, if there are two earthquakes of the same magnitude, each magnitude will count. For example, if the top 4 magnitudes are {7, 6.9, 6.9, and 6}, the 4th largest is 6.

6.4

Correct

This is the magnitude of the earthquake 180km SE of Gizo, Solomon Islands, which is the 6th in the list.

8. Run your program with the earthquake input file "quiz2.atom". There is a commented out line of code in the setUp method in EarthquakeCityMap that you can uncomment to make this happen:

1 / 1 point

```
earthquakesURL = "quiz2.atom";
```

Call sortAndPrint as appropriate to answer the following question.

What is the strongest magnitude that is repeated *three or more times* in the data set?

5.9

Correct

There are 4 earthquakes in this data set with a magnitude of 5.9.

9. In the programming assignment, we asked you to notice and reflect on at least one difference between your code for module 5 and our starter code for module 6 (i.e. our solutions for module 5). Describe that difference and what you noticed about it here. Which solution did you like better, or were they just different?

1 / 1 point

Nothing

Correct

Reading others' code and comparing others solutions to yours is a great way to learn more about code design and algorithmic approaches. If you found that you liked the way we did something better than the way you did it, you can keep that in mind for the future, or change your code to reflect it. If you liked your way better, that's great and make sure you can justify why.

10. How long, total, did you spend on this programming assignment, to the nearest hour? Include only the time you were actively working on the programming assignment including time you spent watching support videos or re-watching videos specifically because you needed help on the assignment.

1 / 1 point

1 hrs

Correct

Thank you for your response.