ADVANCED TOPICS 12 NAME PERIOD

Sort, Search, and BigO 15 PTS

*** After finishing the test, put all your answers on the last page. ***

Questions 1 and 2 refer to the incomplete definition of the `SelectionSorter` class below.

```
public class SelectionSorter
{
     public SelectionSorter(int[] anArray)
     {
     a = anArray;
     }

     public void sort()
     {
          for (int i = 0; I < a.length - 1; i++)
          {
               int minPos = minimumPosition(i);
               swap(minPos,i);
          }
     }

     //Returns the position of the smallest element
     //in a[from]... a[a.length - 1]
     private  int minimumPosition (int from) {...}

     //Exchanges a[i] with a[j]
     swap (int i, int j) {...}

     private int[] a;
}
```

The following segment of code is executed.

```
SelectionSorter sorter = new SelectionSorter(a);
sorter.sort();
```

Consider arrays initialized with the values below.

I. `{2, 3, 4, 5, 6}`
II. `{6, 5, 4, 3, 2}`
III. `{2, 6, 1, 5, 4}`

1. **Which of the following statements regarding the number of comparisons made to sort the array is true?**

a. I takes fewer comparisons than both II and III.
b. II takes fewer comparisons than both I and III.
c. III takes fewer comparisons than both I and II.
d. I and II require the same number of comparisons, which is different from the number of comparisons required to sort III.
e. I, II, and III require the same number of comparisons

2. **Consider the `SelectionSorter` method `sort` on the previous page. Which of the following statements is true at the *beginning* of each iteration of the loop??**

I. For all `j`, such that `0 <= j < i`, `0 <= a[j] <= a[i]`.
II. For all `j`, such that `0 <= j < i`, `a[j]` is in its final position in the array.
III. For all `j`, such that `0 <= j < i`, `a[0]...a[j]` is sorted.

a. I only
b. II only
c. III only
d. II and III only
e. I, II, and III

3. **Which of the following sorts has a partition method that uses a pivot location?**

a. merge sort
b. selection sort
c. quick sort
d. bubble sort
e. insertion sort

4. **What is the bigO of the code below?**

```
int n = //user input
for(int i=0; i<n; i++){
   for(int j=0; j<n; j++){
      System.out.println(i*j);
   }
}
```

a. O(N)
b. O(1)
c. O(N * log2 N)
d. O(Log2N)
e. O(N*N)

5. **Which of the these algorithms has a O(1) best case runtime and a O(N) worst case runtime?**

a. Binary Search
b. Linear Search
c. Selection Sort
d. Insertion Sort
e. Quick Sort

6. **Which of the these algorithms has a O(N*Log2N) best case runtime and a O(N*N) worst case runtime?**

a. Binary Search
b. Linear Search
c. Selection Sort
d. Insertion Sort
e. Quick Sort

7. **Which of the following reserved words would correctly fill < blank 1 > ?**

```
public static void sortOne( Comparable[] list )
{
   for(int i=0; i<list.length-1; i++)
   {
      int min = i;
      for(int j=i+1; j<list.length; j++)
      {
         if(list[j]. < blank 1 > (list[min])<0)
          min = j;
      }
      if( min != i)
      {
        Comparable temp = list[min];
        list[min] = list[i];
        list[i] = temp;
      }
   }
}
```

a. CompareTo
b. int
c. Comparable
d. String
e. Object

8. **Which of the following statements about searching is <u>not</u> true?**

a. The sequential search examines all values in an array until it finds a match or until it reaches the end.
b. A binary search is generally more efficient than a sequential search.
c. A binary search can only be used to search for an item in a sorted array.
d. A sequential search generally takes more comparisons than a binary search.
e. A binary search is always faster than a sequential search.

9. **Which of the following sorts splits data into smaller lists, sorts the smaller lists, and then combines all of the sorted smaller lists back into one big list?**

a. merge sort
b. radix sort
c. quick sort
d. selection sort
e. heap sort

10. **What would be the value of list after four passes of the `help()` method?**

```
public void sort( int[] list, int front, int back)
{
    int mid = (front+back)/2;
    if( mid==front) return;
    sort(list, front, mid);
    sort(list, mid, back);
    help(list, front, back);
}

private void help(int[] list, int front, int back)
{
   int[] temp = new int[back-front];
   int i = front, j = (front+back)/2, k =0;
   int mid =j;
   while( i<mid && j<back)
   {
     if(list[i]< list [j])
        temp[k++]= list[i++];
     else
        temp[k++]= list[j++];
   }

   while(i<mid)
      temp[k++]= list[i++];
   while(j<back)
      temp[k++]= list[j++];
   for(i = 0; i<back-front; ++i)
      list[front+i]=temp[i];
}
```

//code in the main

```
int[] list = {39,6,11,23,18,3,20,5,57};
sort(list, 0, list.length);
```

a. 6 39 11 23 18 3 20 5 57
b. 6 11 23 39 3 18 20 5 57
c. 6 11 23 39 3 18 5 20 57
d. 6 11 23 39 18 3 20 5 57
e. 6 39 11 23 18 3 20 5 57

.. `funSort()` **will put the items in ray in what type of order?**

```
public static void funSort( int[] ray  )
{
   for(int i=0; i< ray.length-1; i++)
   {
        int max = i;
        for(int j = i+1; j < ray.length; j++)
        {
            if( ray[j] > ray[max] )
                 max = j;
        }
        if( max != i){
               int temp = ray[max];
           ray[max] = ray[i];
           ray[i] = temp;
        }
    }
}
```

a. ascending
b. descending
c. random
d. median
e. returned order will be same as original order

12. **If ray is storing 10 elements, what is the maximum number of swaps** `funSort()` **could make in order to put the elements in order?**

```
public static void funSort( int[] ray  )
{
   for(int i=0; i< ray.length-1; i++)
   {
        int max = i;
        for(int j = i+1; j < ray.length; j++)
        {
            if( ray[j] > ray[max] )
                 max = j;
        }
        if( max != i){
               int temp = ray[max];
           ray[max] = ray[i];
           ray[i] = temp;
        }
    }
}
```

a. 8
b. 9
c. 10
d. 11
e. 7

13. **Consider the following instance variable and incomplete method. The method `findInsertLocation` should return the position in the `ArrayList` where the new word should be added in order to maintain the sorted ascending order of the list.**

```
private ArrayList<String> list;

public int findInsertLocation( String word )
{
   int loc = 0;

      /* code */

   return loc;
}
```

Which of the following code segments shown below could be used to replace /* code */ so that `findInsertLocation` will work as intended?

I.
```
for( String s : list )
{
   if( s > word )
      return loc;
   loc++;
}
```

II.
```
for( String s : list )
{
   if( s.compareTo( word ) > 0 )
      return loc;
   loc++;
}
```

III.
```
for( String s : list )
{
   if( s.compareTo( word ) < 0 )
      return loc;
   loc++;
}
```

a. I only
b. II only
c. II and III only
d. I and II only
e. I and III only

If `funSearch()` is passed a list that contains 100 integers in sorted order, how many checks will `funSearch()` have to make before it determines that the item it is searching for is NOT in the list ?

```
public static int funSearch( int[] list, int val ){
   int len=list.length;
   int bot=0, top=len-1;
   int middle=0;
   while( bot <= top )
   {
      middle = (bot+top)/2;
      if( list[middle] == val )
         return middle;
      else
         if( list[middle] > val )
            top=middle-1;
         else
            bot=middle+1;
   }
   return -1;
}
```

a. 5
b. 6
c. 7
d. 8
e. 9

15. **Assuming <blank 1> is filled correctly, what sort is sortOne()?**

```
public static void sortOne( Comparable[] list ){{
   for(int i=0; i<list.length-1; i++)
   {
      int min = i;
      for(int j=i+1; j<list.length; j++)
      {
         if(list[j].  <blank 1>  (list[min])<0)
          min = j;
      }
      if( min != i)
      {
        Comparable temp = list[min];
        list[min] = list[i];
        list[i] = temp;
      }
   }
}
```

a. bubble sort
b. selection sort
c. insertion sort
d. binary search
e. merge sort

Answer Page

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
a	d	c	e	b	e	a	e	a	c	b	c	b	b	b

b b c