

CSci 161 Midterm Exam, Fall 2009
Score: #MC correct/30 + #SA/30 + #C/15 = _____

Name _____

Multiple Choice Questions: Circle the best choice. (# correct = _____)

1. In object-oriented programming, an object
 - a. is a class
 - b. may contain data and methods
 - c. is a program
 - d. may contain classes

2. Inserting an item into an unordered array
 - a. takes time proportional to the number of items in the array
 - b. requires multiple comparisons
 - c. requires shifting other items to make room
 - d. takes the same time, no matter how many items are in the array

3. Ordered arrays, compared with unordered arrays, are
 - a. much quicker at deletion
 - b. quicker at insertion
 - c. quicker to create
 - d. quicker at searching

4. The maximum number of elements that must be examined to complete a binary search in an array of 200 elements is
 - a. 200
 - b. 8
 - c. 1
 - d. 13

5. In an unordered array, allowing duplicates
 - a. increases execution times for insert, delete and search operations
 - b. increases search times in some situations
 - c. increases insertion times
 - d. decreases insertion times

6. If a method does not intend to handle an exception that could occur while it is executing, it can
 - a. catch that exception
 - b. try that exception
 - c. retry that exception
 - d. throw that exception

7. If a queue contains N elements when a dequeue operation is performed, the execution time for that operation is expected to be
 - a. proportional to N
 - b. proportional to $\log(N)$
 - c. proportional to N^2
 - d. a constant

8. If an unsorted array contains N elements when a bubblesort operation begins, the execution time for that sort operation is expected to be

- a. proportional to N
- b. proportional to $\log(N)$
- c. proportional to N^2
- d. a constant

9. The readDouble() method is one of the methods for objects of type

- a. FileInputStream
- b. **DataInputStream**
- c. ObjectInputStream
- d. InputStreamReader

10. When an input operation encounters end-of-file

- a. an exception is always thrown
- b. a nonsense value is always returned
- c. a JOptionPane dialog window pops up
- d. **none of the above**

Short Answer/Fill in the Blank Exercises: (# correct = _____)

1. In Java, boolean, long and byte are examples of primitive types

2. What are the three important issues to address (checklist items) when designing or coding a recursive algorithm?

- Identify a simple version of the problem (a base case) to be solved without using recursion
- Be sure that each recursive call is used to solve a simpler version of the problem than the one the caller is trying to solve
- Figure out how to use the solutions of the simpler problem instances to solve the more complex instance

3. An array containing N elements is to be searched to see if it contains a particular target value. What is the time complexity of the most efficient algorithm for performing that search? Does it make any difference if the array is sorted or not?

if sorted, binary search with time complexity $O(\log(N))$ can be used
if unsorted, linear search with time complexity $O(N)$ must be used

4. Suppose an algorithm has time complexity $T(N)=O(N^2)$ when used to solve a problem of size N. What effect does doubling the problem size have on the execution time of this algorithm?

$(2N)^2 = 4N^2$, so doubling the problem size can quadruple the execution time

5. The base 2 logarithm of 32 is 5.

6. A Java class definition might contain *fields*, *constructors* and *methods*. Briefly describe the purpose of each of these components of a class definition.

fields are variables used to store an object's data

methods are subroutines which contain executable code implementing the operations on objects of this type

constructors, which contain executable code, are called when an instance of the object type is created, to properly initialize the fields of that instance (and possibly do other work as well)

7. The array [45, 38, 21, 17, 66, 25, 19, 42, 49] is to be sorted using bubblesort. What does the array look like after the first bubblesort pass?

[38, 21, 17, 45, 25, 19, 42, 49, 66] or [17, 45, 38, 21, 19, 66, 25, 42, 49]

8. The array [45, 38, 21, 17, 66, 25, 19, 42, 49] is to be sorted using insertion sort. What does the array look like after the first insertion sort pass?

[38, 45, 21, 17, 66, 25, 19, 42, 49]

The sublist consisting of the first two items is now sorted. Nothing else was moved.

9. The array [45, 38, 21, 17, 66, 25, 19, 42, 49] is to be sorted using selection sort. What does the array look like after the first selection sort pass?

[45, 38, 21, 17, 49, 25, 19, 42, 66] or [17, 38, 21, 45, 66, 25, 19, 42, 49]

ONLY ONE swap should have been made

10. Explain, abstractly, how a priority queue works. I.e., name and describe the priority queue operations (methods).

the insert method adds an item to the priority queue (in some unspecified manner)
the remove method removes and returns the highest priority item from the priority queue
other useful methods could include front, isEmpty, isFull.

Coding Exercises: (# correct = _____)

1. Write Java code to search an unordered array of integers named `thisArray` for a certain value named `target`, assuming that a variable named `nItems` contains the number of data values stored in that array. If found, display the index where the target value was found, else display -1 to indicate the target was not found.

```
int [] thisArray;
int nItems;
int target;
```

<pre>int i=0; int index=-1; while(index== -1 && i<nItems) { if (thisArray[i]==target) { index=i; } i++; } System.out.println("index:"+index);</pre>	<pre>int index=-1; for (int i=0; i<nItems; i++) { if (thisArray[i]==target) { index=i; break; } } System.out.println("index: "+index);</pre>
---	--

2. The *Stack* is an abstract data type with certain properties or capabilities. Code a Java interface that describes the operations of this abstract data type. Write a 1-line comment for each of the method signatures, describing the action it performs.

```
public interface Stack {

    // add an item to the top of the stack
    public boolean push(Object x);

    // remove and return the most recently added item
    public Object pop();

    // return (with removing) the most recently added item
    public Object top();

    // other useful operation could be specified, such as
    // isFull(), isEmpty()
}
```

3. Write the Java code to compute $n!$ using recursion.

```
public int factorial(int n) {  
    if (n==0 || n==1) return 1;  
    return n*factorial(n-1);  
}
```

4. Write Java code that will display the number of command line parameters received by the main method and then, if that number is 3 or greater, display the value of the next-to-last command line parameter.

```
public static void main(String[] args) {  
    System.out.println("there are " + args.length +  
        " command line parameters");  
    if (args.length > 2) {  
        System.out.println("the next-to-last parameter is "+  
            args[args.length-2]);  
    }  
}
```

5. Write Java code for a public method named *powers* which is passed two positive integers, *a* and *b*, as parameters. This method is supposed to return an array of integers of length $b+1$, where the element in that array with index j has the value a^j for $j=0, 1, \dots, b$. Note that your method needs to create the array as well as to initialize each of the values in the array.

```
public int[] powers(int a, int b) {  
    int[] myArray = new int[b+1];  
    myArray[0]=1;  
    for (int i=1; i<myArray.length; i++) {  
        myArray[i] = a * myArray[i-1];  
    }  
    return myArray;  
}
```