This Test has 44 attempts. For information on editing questions, click **More Help** below.

Test Canvas: Mid-term Exam- Requires Respondus LockDown Browser

The Test Canvas lets you add, edit, and reorder questions, as well as review a test. More Help

Question Settings

You can edit, delete, or change the point values of test questions on this page. If necessary, test attempts will be regraded after you submit your changes.

Description	This is the first part of your mid-term exam. It is "Closed Book" which means you are allowed one sheet of notes (single-sided if printed, otherwise double-sided). Do this part of the exam first and then move on to the HackerRank portion. The total time for the exam is 90 minutes, but Blackboard will auto-submit your code after 65 minutes.
Instructions	
Total Questions	7
Total Points	72
Number of Attempts	44

Select: All None | Select by Type: - Question Type -

Delete and Regrade | Points | Update and Regrade | Hide Question Details

1. **Multiple Choice: Java hashCode method: In which of the following cases ...** Points: 4

Question	In which of the following cases should you override *hashCode()* in your class?
Answer	You are going to create a list of objects of your class with list size greater than 4 billion, for the purpose of storing all of them in a file.
	✓ You are going to use *HashSet* to create a collection of instances of your class.
	You have not also overridden the *equals()* method of the *Object* class in your class.
	Your class implements the *Comparable* interface.

2. **Multiple Choice: The Dictionary Principle: A dictionary** Points: 10

is something where intel...

Question	A dictionary is something where intellectual work is done up front (the sort) in order to save work later on (the searches). In this question, you are to compare the work done (in terms of total compares) for M lookups on a dictionary object of N elements in two separate cases. Case 1 is where there is no sort before the lookups; Case 2 is where a sort requiring $N^2/4$ compares precedes the lookups. What is the break-even value of M between cases 1 and 2 when N = 1000 (rounded to the nearest integer) Consider the average case while thinking about sorting and searching algorithm complexity.
Answer	✓ A. 510 B. 520 C. 1010 D. 480

☐ 3. **Multiple Choice: Merge sort cutoff: In merge sort, as the number of ...**

Points: 10

Question	In merge sort, as the number of elements to sort in a sub-array decreases (halving each time), you have to consider when is an appropriate time to switch to insertion sort. Your team members make the following arguments. Pick the one you think is best.
Answer	✓ A. You should perform an experiment, timing the results of different cutoffs, using many different initial conditions (random, partially-sorted, etc.), each with many repetitions. B. You should cut to insertion sort whenever you get within one of a stack overflow. C. Continuing with merge sort will take N lg N compares, while switching to insertion sort will take N^2/2 compares. You should cut over to insertion sort whenever N is less than 8. D. Continuing with merge sort will take N lg N compares, while switching to insertion sort will take N^2/4 compares. You should cut over to insertion sort whenever N is less than 16. E. Don't bother. The additional test that you make on every iteration will outweigh any savings you might get for the small arrays.

☐ 4. **Multiple Answer: Quick sort: In quicksort, the average number...**

Points: 12

Question	

In quicksort, the average number of compares to sort an array of N elements is *2 N ln N*, while the best (fewest) number of compares is *N lg N*.

Mergesort, however, requires between *1/2 N lg N* and *N lg N* compares. That's to say that the worst-case number of compares for mergesort is equal or better than the best-case number for quicksort.

How is it, then, that quicksort tends to be faster than mergesort when sorting, say, an array of *N doubles*.

Answer

- A. Quicksort typically involves fewer array accesses than mergesort.
- B. The assertion is in error. *2 N ln N* is actually a smaller number than *N lg N*.
- C.

  The cost of copying the array to the auxiliary array is significant and this contributes to making mergesort slower
- D.

  The number of compares is not the only factor. The number of cache misses is significant, the cost of randomizing the array is significant, cost of swapping and/or copying is important, the number of pointers with their increment/decrement operators is also significant. In fact, comparing the performance of two different sort algorithms is quite complex.

5. Multiple Answer: Queue: Which of the following statements is ...

Points: 12

Question

Which of the following statements is true with respect to a queue?

Answer

- A. A queue is a first-in, first-out (FIFO) data structure.
- B. A fixed-size array is a suitable internal data structure for a queue.
- C. A doubly-linked list is a suitable internal data structure for a queue.
- D.

  A singly-linked list, together with a pointer to the last element, is a better internal data structure than a doubly-linked list because it uses up less space and is slightly more efficient.
- E. In Java, a queue must throw an exception if the *dequeue* method is invoked on an empty queue.

6. Multiple Answer: Complexity of algorithms: Which of the following statements are...

Points: 12

Question

Which of the following statements are true about the complexity of algorithms?

Answer

- A.

  An *optimal* algorithm is one where the upper bound for the complexity of the algorithm is, to within a *constant* factor, equal to the lower bound for the problem.

B. Tilde ("~") notation and Big Oh (**O**) are the same thing.

C.

Improving an algorithm's order of growth from ~ N^2 to ~ $N \log N$ [for example, replacing insertion sort by merge sort] is generally more significant than going from ~ $N \log N$ to ~ N [for example, the improvement between doing a quick sort and a quick select].

D.

There is no significant advantage in reducing the order of growth of an algorithm from ~ $\log N$ to ~ $\log \log N$.

7. **Multiple Answer: Huffman Coding: Which of the following statements are...**

Points: 12

Question	Which of the following statements are true with respect to Huffman Coding?
Answer	A. A Huffman code is unique to the particular set of messages for which it is designed.
	B. A Huffman codon (the specific code for a letter in a message) is self-terminating, that's to say there is no special symbol in the code to separate letters.
	C. Morse code is an early form of a Huffman code.
	D. A Huffman code is designed to minimize the expected total length of any set of actual messages.

Select: All None | Select by Type: - Question Type -

Delete and Regrade | Points | Update and Regrade | Hide Question Details

← OK