Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

1. Insertion sort is an example of divide and conquer? **1 / 1 point**

True [✓ False]

Correct
That's correct. Insertion sort processes each element in relation to its surrounding elements until the data is eventually sorted.

2. Given an array of 6 numbers `[6,8,19,48,9,90]` and applying **insertion sort,** how many swaps must occur before the array is sorted? **1 / 1 point**

6 [✓ 2] 4

Correct
That's correct. The array is mostly ordered so only have to swap 9 twice, first with 48, then with 19.

3. What time complexity is required to do a linear search? **1 / 1 point**

[✓ `O(n)`] `O(1)` `((log (n))`

Correct
That's correct. A linear search requires that you do a search of every item. So it will take n (the number of items) time to search.

4. Why do we need Big-O notation to evaluate our programs? **1 / 1 point**

- Because sorting requires that things are moved around to save space.
- [✓ Because measuring time is relative to a person's computer, so a relative metric is required.]
- Because sorting is complicated, and we need a complicated metric.

Correct
That's correct. A relative metric is required to measure time.

5. What is parallelization? **1 / 1 point**

- It is about writing your code in one go.
- [✓ It is about running code at the same time in threads or on separate computers.]
- It is about calling functions repetitively until they have achieved a base case.

Correct
That's correct. You have successfully identified a brief definition of parallelization.

6. Why would you decide to use recursion? **1 / 1 point**

- [✓ It lends itself well to a divide and conquer approach.]
- Recursion reduces the pressure on the compiler by making less stack calls.
- It looks cool and makes your code seem more intelligent.

Correct
That's correct. Recursion works well with the divide and conquer approach.

7. Why does Memoization work well with dynamic programming?

1 / 1 point

- [✓ It requires less compiling because it stores previous results, reducing the load on the CPU.]
- It takes up less space in the hard drive.
- Because it takes a lot of memory to run some programs and memoization allows you to store data in smaller sizes.

Correct
That's correct. Dynamic programming utilizes memoization because it stores the results of computations, meaning the computations don't have to be repeated.

8. How are the principles of dynamic programming and greedy algorithms at odds with one another?

1 / 1 point

- Because dynamic programming will react with more agility to a program, while the greedy approach will be slower and more self-centered.
- [✓ The principle of dynamic programming is to exhaustively compute the best solution, while a greedy approach will favor take the immediate best option.]
- The greedy algorithm will use up CPU by monopolizing resources.

Correct
That's correct. With dynamic programming, you can find the most best solution, whereas greedy algorithms have a specific process.

9. Why is a binary search conducted in `O(log n)` time?

1 / 1 point

- It is not, it is conducted in `O(n)`.
- [✓ Regardless of the size of the input, at every step the number of calculations is halved.]
- Because as it searches it sorts the elements.

Correct
That's correct. Log n means that it is not instantaneous access but it rapidly reduces the lookup space.

10.

1 / 1 point

```
def fibonacci(number)
  if number < 2
    number
  else
    fibonacci(number - 1) + fibonacci(number - 2)
  end
end
```

In the Fibonacci pseudocode above how many recursive instances can be seen?

- 0
- 1
- [✓ 2]

Correct
That's correct. The algorithm is being called on the last, and second to last number on the series.