2026 Reliable Foundations-of-Computer-Science–100% Free Testking | Reliable WGU Foundations of Computer Science Dumps Free

2026 Latest Exams4Collection Foundations-of-Computer-Science PDF Dumps and Foundations-of-Computer-Science Exam Engine Free Share: https://drive.google.com/open?id=1zsbURBmCGf_LUEbk7VarWv5KcYIg2sA7

You can download Exams4Collection WGU Foundations-of-Computer-Science PDF dumps file on your desktop computer, laptop, tab, or even on your smartphone. Just download the Foundations-of-Computer-Science PDF questions file after paying affordable Prepare for your WGU Foundations of Computer Science (Foundations-of-Computer-Science) exam questions charges and start WGU Foundations of Computer Science (Foundations-of-Computer-Science) exam preparation anytime and anywhere.

Our staff is suffer-able to your any questions related to our Foundations-of-Computer-Science test guide. If you get any suspicions, we offer help 24/7 with enthusiasm and patience. Apart from our stupendous Foundations-of-Computer-Science latest dumps, our after-sales services are also unquestionable. Your decision of the practice materials may affects the results you concerning most right now. Good exam results are not accidents, but the results of careful preparation and high quality and accuracy materials like our Foundations-of-Computer-Science practice materials.

>> Foundations-of-Computer-Science Testking <<

Reliable Foundations-of-Computer-Science Dumps Free, Foundations-of-Computer-Science Pdf Exam Dump

Only to find a way to success, not to make excuses for failure. Exams4Collection's Foundations-of-Computer-Science exam certification training materials include Foundations-of-Computer-Science exam dumps and answers. The data is worked out by our experienced team of IT professionals with their own exploration and continuous practice. Exams4Collection's Foundations-of-Computer-Science Exam Certification training materials have high accuracy and wide coverage. It will be a grand helper that will accompany you to prepare for Foundations-of-Computer-Science certification exam.

WGU Foundations of Computer Science Sample Questions (Q52-Q57):

NEW QUESTION # 52
Which action is taken if the first number is the lowest value in a selection sort?

- A. The first number is duplicated.
- B. The first number is increased by one.
- C. It swaps the selected element with the first unsorted element.
- D. It swaps the selected element with the last unsorted element.

Answer: C

Explanation:
Selection sort works by maintaining a boundary between a sorted prefix and an unsorted suffix. On each pass, the algorithm finds the smallest value in the unsorted portion and places it into the first position of that unsorted portion (which is also the next position in the sorted prefix). This is usually done by swapping the element at the minimum's index with the element at the boundary index (the "first unsorted element"). That description matches option D.
If the first element of the unsorted portion is already the smallest, then the minimum's index equals the boundary index. In textbook implementations, the algorithm may still execute a swap operation, but it becomes a swap of an element with itself (a no-op), leaving the array unchanged. Many implementations include a small optimization: perform the swap only if the minimum index differs from the boundary index.
Either way, conceptually the "action taken" by selection sort is still "swap the selected minimum into the first unsorted position," which is exactly what option D states.
Options A and B are unrelated to sorting; selection sort never increases or duplicates values. Option C is incorrect because selection sort swaps the minimum with thefirstunsorted element, not the last. After the swap (or no-op), the sorted region grows by one element, and the algorithm repeats from the next boundary position.
This logic is fundamental for understanding how selection sort ensures correctness: after pass i, the smallest i+1 elements are fixed in their final positions.

NEW QUESTION # 53
Which line of code below contains an error in the use of NumPy?

- A. arr = np.array([3, 2, 0, 1])
- B. print(wgu_list)
- C. wgu_list = np.quicksort(arr)
- D. import numpy as np

Answer: C

Explanation:
The NumPy library provides arrays and efficient numerical operations, including sorting. However, NumPy doesnotprovide a function named np.quicksort. That is the API misuse in the code, making option A the correct answer. In NumPy, sorting is commonly performed using np.sort(arr) (which returns a sorted copy) or arr.sort() (which sorts in-place). If a specific algorithm is desired, NumPy exposes it through the kind parameter, such as np.sort(arr, kind="quicksort"), kind="mergesort", or kind="heapsort". Textbooks present this as a typical design: a single sorting interface with selectable strategies, rather than separate top-level functions per algorithm name.
Option C is correct and necessary: import numpy as np is standard convention. Option B is also correct:
printing a variable is valid assuming it exists. Option D, written as arr = np.array([3, 2, 0, 1]), is valid NumPy usage for constructing a 1D array from a Python list.
A subtle point taught in scientific computing courses is that library APIs matter as much as syntax: you can write perfectly valid Python that still fails if you call a function that the library does not define. In this case, the fix is to replace np.quicksort(arr) with np.sort(arr) or np.sort(arr, kind="quicksort") depending on whether you need to specify the algorithm.

NEW QUESTION # 54
Which principle can be used to implement an algorithm to calculate factorial or Fibonacci sequence?

- A. Recursion programming
- B. Iterative programming
- C. Procedural programming
- D. Object-oriented programming

Answer: A

Explanation:
Factorial and Fibonacci are classic examples used to teachrecursion, a technique where a function solves a problem by calling itself on smaller subproblems. The key requirement for recursion is (1) abase casethat stops further calls and (2) arecursive casethat reduces the problem size. For factorial, the definition is (n! = n
\times (n-1)!) with base case (0! = 1) (or (1! = 1)). For Fibonacci, (F(n) = F(n-1) + F(n-2)) with base cases (F (0)=0) and (F(1)=1). These mathematical definitions map directly into recursive code, which is why textbooks frequently introduce recursion using these sequences.
While factorial and Fibonacci can also be computed iteratively, the question asks for the principle that can be used to implement such algorithms, and recursion is the canonical textbook answer. Recursion also connects to important CS topics: call stacks, activation records, and divide-and-conquer problem solving.
Option A ("procedural programming") and option D ("object-oriented programming") are broader paradigms rather than the specific technique used in the classic implementations. Option B ("iterative programming") is a valid alternative approach, but the standard instructional principle highlighted for these particular examples is recursion. Textbooks also note that naive recursive Fibonacci is inefficient (exponential time) unless optimized with memoization or converted to an iterative or dynamic programming approach.

NEW QUESTION # 55
What is the time complexity of a binary search algorithm?

- A. O(n)
- B. O(n