Given below are four functions that try to calculate the factorial of N. Which of these are correct?

Select one or more:

a.
factorial(N)

```
if N == 0:
    return 1
else:
    return factorial(n-1) * n
```

b.
factorial(N)

```
if N == 0 or N == 1:
    return 1
```

Regenerate response

c.
factorial(N)

```
if N == 1:
    return 1
else:
    return factorial(n-1) * n
```

d.
factorial(N)

```
    return factorial(n-1) * n
```

e.
factorial(N)

Regenerate response

The correct functions for calculating the factorial of N are b and c.

Function a is incorrect because it does not handle the case where N is negative. If N is negative, the function will enter an infinite recursion.

Function d is incorrect because it does not have a base case to terminate the recursion. It will cause a stack overflow error.

Function e is incorrect for the same reason as function d. It does not have a base case to terminate the recursion.

Function b is correct because it handles the base cases of N equals 0 or 1, which returns 1. It also correctly calculates the factorial for N greater than 1 by calling itself recursively.

Function c is also correct because it correctly calculates the factorial of N by calling itself recursively until it reaches the base case where N equals 1.

Therefore, the correct functions for calculating the factorial of N are b and c.

Regenerate response

a.
factorial(N)

```
if N == 0:
    return 1
else:
    return factorial(n-1) * n
```

b.

```
factorial(N)
if N == 0 or N == 1:
    return 1
else:
```

Regenerate response


```
factorial(N)
if N == 1:
    return 1
else:
    return factorial(n-1) * n
```

d.

```
factorial(N)
    return factorial(n-1) * n
```

e.

```
factorial(N)
    return factorial(n-1)
```

Regenerate response

Function a is incorrect because it does not handle the case where N is negative. If N is negative, the function will enter an infinite recursion.

Function d is incorrect because it does not have a base case to terminate the recursion. It will cause a stack overflow error.

Function e is incorrect for the same reason as function d. It does not have a base case to terminate the recursion.

Function b is correct because it handles the base cases of N equals 0 or 1, which returns 1. It also correctly calculates the factorial for N greater than

Algorithm(s) which use divide and conquer approach

a.
Binary search

Regenerate response

c.
Merge sort

d.
Selection Sort

The algorithms that use the divide and conquer approach are:

a. Binary search - This algorithm uses the divide and conquer approach to search for a target

value in a sorted array by repeatedly dividing the array in half and discarding the half that does not contain the target value.

b. Merge sort - This algorithm uses the divide and conquer approach to sort an array by repeatedly dividing the array in half, sorting each half recursively, and then merging the sorted halves back together.

c. Quick sort - This algorithm uses pproach to sort an array by selecting a pivot element, partition subarrays based on the pivot

Regenerate response

In a little game, a computer is going to randomly select an integer from 1 to 2000. You'll keep guessing numbers until you find the computer's number, and the computer will tell you each time if your guess was too high or too low. How many quesses you need atmost in your worst case scenario given you use an optimal strategy?

If we use an optimal strategy, we can use binary search to guess the number. With each guess, we can eliminate half of the remaining numbers.

Let's say we start by guessing the middle number, which is 1000. If the computer tells us that the number is too high, we can eliminate all numbers from 1000 to 2000. If the computer tells us that the number is too low, we can eliminate all numbers from 1 to 1000.

In the worst case scenario, we would need to guess the number in 11 steps or less. This is because 2^11 is the smallest power of 2 that is greater than 2000. With each guess, we can eliminate half of the remaining nu s, we can eliminate all but one number, which must be the computer's number.

Regenerate response

Given the following method declaration, what will redo(82, 3) return?

```
public static int redo(int i, int j)
{
  if (i==0)
    return 0;
  else
    return redo(i/j, j)+1;
}
```

Here '/' is integer division.

The redo(82, 3) method call will return 4.

Here's why:

First call: redo(82, 3)

redo(82/3, 3) + 1 = redo(27, 3) + 1

Regenerate response

Third call: redo(9, 3)

redo(9/3, 3) + 1 = redo(3, 3) + 1

Fourth call: redo(3, 3)

redo(3/3, 3) + 1 = redo(1, 3) + 1

Fifth call: redo(1, 3)

redo(1/3, 3) + 1 = redo(0, 3) + 1

Sixth call: redo(0, 3)

Since i is 0, the method returns 0.

Therefore, redo(82, 3) returns 4.

Regenerate response

Consider the following functions,

ChatGPT Mar 14 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

```
    return True
else:
    return func2(N-1)


func2(N)
if N == 0:
    return False
else:
    return func1(N-1)
```

Regenerate response

ChatGPT Mar 14 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

In these functions, two or more functions are defined in terms of each other, in a way that they call each other in a circular manner. In the given example, func1 calls func2, and func2 calls func1.

Mutually recursive functions are a type of indirect recursion, where the recursion is achieved through a chain of function calls, rather than a function calling itself directly.

Worst case time complexity of Merge Sort

a.
O(n log(n))

b.
O(n)

Regenerate response

ChatGPT Mar 14 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

d.
O(n2)

The worst case time complexity of Merge Sort is O(n log(n)).

In Merge Sort, the array is recursively divided into halves until each subarray has only one element, and then the subarrays are merged together in sorted order.

In the worst case scenario, each element in the array needs to be compared with every other element during the merge step. This means that the number of comparisons required is proportional to n log(n), where n is the size of the array.

Therefore, the worst case time complexity of Merge Sort is O(n log(n)).

Regenerate response

ChatGPT Mar 14 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.