Started on	Saturday, 3 December 2016, 3:42 PM
State	Finished
Completed on	Saturday, 3 December 2016, 3:44 PM
Time taken	2 mins 36 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question 1

Correct Mark 1.00 out of 1.00

Which one of the following is not naturally recursive?

Select one:

- a. Sum of N numbers ✓
- b. Tower of Hanoi
- c. Factorial
- d. Fibonacci

Your answer is correct.

The correct answer is: Sum of N numbers

Question 2

Correct Mark 1.00 out of 1.00

Which one of the following functions have a correct base case?

Select one:

a. def Factorial(n):
if (n == 0): return 1
return n * Factorial (n - 1) ✓

b. def Factorial(n):
if (n > 0): return 1
return n * Factorial (n - 1)

c. def Factorial(n):
if (n == 0): return n
return n * Factorial (n - 1)

d. def Factorial(n):
if (n == 0): return n-1
return n * Factorial (n - 1)

Your answer is correct.

The correct answer is: def Factorial(n):
if (n == 0): return 1
return n * Factorial (n - 1)

Question 3

Correct Mark 1.00 out of 1.00

What is wrong with the following recursive function?

```
def Fibonacci(n):
    if (n == 0): return 0
    if (n == 1): return 1
    return Fibonacci(n - 1) + Fibonacci(n - 2)
```

Select one:

- a. Recursive part should be

  ```
  return Fibonacci(n) + Fibonacci(n - 1)
  ```

- b. There cannot be two base cases
- c. It calculates only even Fibonacci numbers
- d. Nothing ✓

Your answer is correct.

The correct answer is: Nothing

Question 4

Correct Mark 1.00 out of 1.00

Recursion is not always the best solution.

Select one:

- True ✓
- False

The correct answer is 'True'.

Question 5

Correct Mark 1.00 out of 1.00

Which one of the following is a recursive implementation of binary search?

Select one:

- a. def BinarySearchRecursive(values,target):
  #base case
  if (len(values) == 0): return False

  #recursion
  mid = len(values) // 2
  if (target == values[mid]):
  return True
  if (target > values[mid]):
  return BinarySearchRecursive(values[:mid],target)
  elif (target < values[mid]):
  return BinarySearchRecursive(values[mid+1:],target)
- b. def BinarySearchRecursive(values,target):
  #base case

```
if (len(values) == 0): return False


#recursion
mid = len(values) // 2
if (target == values[mid]):
    return True
if (target < values[mid]):
    return BinarySearchRecursive(values[:mid],target)
elif (target > values[mid]):
    return BinarySearchRecursive(values[mid+1:],target)
```

c.
```
def BinarySearchRecursive(values,target):
#base case
if (len(values) == 0): return False


#recursion
if (target == values[mid]):
    return True
if (target < values[mid]):
    return BinarySearchRecursive(values[:mid],target)
elif (target > values[mid]):
    return BinarySearchRecursive(values[mid+1:],target)
```

d.
```
def BinarySearchRecursive(values,target):
#base case
if (len(values) > 0): return True


#recursion
mid = len(values) // 2
if (target == values[mid]):
    return True
if (target < values[mid]):
```

```
        return BinarySearchRecursive(values[:mid],target)
    elif (target > values[mid]):
        return BinarySearchRecursive(values[mid+1:],target)
```

Your answer is correct.

The correct answer is: def BinarySearchRecursive(values,target):

```
    #base case
    if (len(values) == 0): return False



    #recursion
    mid = len(values) // 2
    if (target == values[mid]):
        return True
    if (target < values[mid]):
        return BinarySearchRecursive(values[:mid],target)
    elif (target > values[mid]):
        return BinarySearchRecursive(values[mid+1:],target)
```