Quiz 4: CSE101 – Introduction to Computers

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID No: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Binary search involves reducing the size of the search area in a sorted array. It does this by: (1 pt)
   1. Comparing an element at the midpoint of the region with the search item and setting the new upper bound of the region to the midpoint if the search item is greater than the element at the midpoint or setting the new lower bound to the midpoint if the item is less than the element at the midpoint.
   2. Comparing an element at the midpoint of the region with the search item and returning false if the search term does not match the element at the midpoint.
   3. Comparing an element at the midpoint of the region with the search item and setting the new lower bound of the region to the midpoint if the search item is greater than the element at the midpoint or setting the new upper bound to the midpoint if the item is less than the element at the midpoint.
   4. None of the above
2. The merge sort algorithm works: (1 pt)
   1. Top-down
   2. Bottom-up
   3. Left-to-right
   4. Right-to-left
3. Slicing on a list allows you to divide elements in a list by the same integer and returns a new list with the resulting values [ True / False ] (1 pt)
4. The quicksort algorithm works:  (1 pt)
   1. Top-down
   2. Bottom-up
   3. Left-to-right
   4. Right-to-left
5. Part of the recursive solution for finding palindromes is to verify that the first and last character in a word match. The next step is to call the routine recursively with an argument that removes those two characters. If the variable *strng* holds the string we are working with, we remove the first and last character with: (1 pt)
   1. strng[-1:1]
   2. strng[:-1]
   3. strng[1:-1]
   4. strng[1:]
6. Recursive routines have code that handle two different cases. Those are the: (1 pt)
   1. Recursive function call and iterative case(s)
   2. Standard case(s) and recursive function call
   3. Base case(s) and recursive function call
   4. Simple case(s) and recursive function call
7. A binary search of a sorted list of 64 elements should take \_\_\_\_\_\_ iterations: (1 pt)
   1. 4
   2. 6
   3. 8
   4. 32
8. The following is an implementation of a method, rindex, which uses a recursive helper method ,rindex\_helper:

def rindex(string, target):

return rindex\_helper(string, target, 0)

def rindex\_helper(string, target, i):

if i >= len(string):

return None

elif string[i] == target:

return i

else:

return rindex\_helper(string, target, i+1)

Why is the helper function needed? (1 pt)

* 1. It is needed since there is no return value and the helper uses an index to track how far into the string we have processed
  2. It is needed since the method is working with strings instead of lists
  3. It is needed to reduce complexity and allow the method to run quicker
  4. It is not needed. It could have been easily written as a single recursive method

1. Consider a situation where you don't have function to calculate power in Python and you need to calculate xn where x can be any number and n is a positive integer. What can be the best possible time complexity of your power function? (1 pt)
   1. O(n)
   2. O(nlogn)
   3. O(loglogn)
   4. O(logn)
2. Which is the most appropriate definition for recursion? (1 pt)
   1. A function that calls itself
   2. A function execution instance that calls another execution instance of the same function
   3. A class method that calls another class method
   4. An in-built method that is automatically called
3. Which of these is false about recursion? (1 pt)
   1. Recursive function can be replaced by a non-recursive function
   2. Recursive functions usually take more memory space than non-recursive function
   3. Recursive functions run faster than non-recursive function
   4. Recursion makes programs easier to understand
4. Fill in the line of code for calculating the factorial of a number. (1 pt)

def fact(num):

if num == 0:

return 1

else:

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. num\*fact(num-1)
  2. (num-1)\*(num-2)
  3. num\*(num-1)
  4. fact(num)\*fact(num-1)

1. Which of the following statements is false about recursion? (1 pt)
   1. Every recursive function must have a base case
   2. Infinite recursion can occur if the base case isn’t properly mentioned
   3. A recursive function makes the code easier to understand
   4. Every recursive function must have a return value
2. What is the output of the following piece of code? (2 pts)

def test(i,j):

if(i==0):

return j

else:

return test(i-1,i+j)

print(test(4,7))

* 1. 13
  2. 7
  3. Infinite loop
  4. 17

1. What is the output of the following code? (2 pts)

l=[]

def convert(b):

if(b==0):

return l

dig=b%2

l.append(dig)

convert(b//2)

convert(6)

l.reverse()

print(l)

* 1. [0, 1, 1]
  2. [1, 1, 0]
  3. 3
  4. Infinite loop

1. What is the output of the following piece of code? (2 pts)

def fun(n):

if (n > 100):

return n - 5

return fun(n+11);

print(fun(45))

* 1. 96
  2. 106
  3. 104
  4. Infinite loop

1. What happens if the base condition isn’t defined in recursive programs? (1 pt)
   1. Program gets into an infinite loop
   2. Program runs once
   3. Program runs n number of times where n is the argument given to the function
   4. An exception is thrown
2. Which of these is not true about recursion? (1 pt)
   1. Making the code look clean
   2. A complex task can be broken into sub-problems
   3. Recursive calls take up less memory
   4. Sequence generation is easier than a nested iteration
3. What is the output of the following piece of code? (2 pt)

def a(n):

if n == 0:

return 0

elif n == 1:

return 1

else:

return a(n-1)+a(n-2)

for i in range(0,4):

print(a(i),end=" ")

* 1. 0 1 2 3
  2. An exception is thrown
  3. 0 1 1 2 3
  4. 0 1 1 2

1. The complexity of mergesort algorithm is ….. (1 pt)
   1. O(n)
   2. O(logn)
   3. O(n2)
   4. O(n logn)
2. Partition and exchange sort is …….. (1 pt)
   1. quick sort
   2. tree sort
   3. heap sort
   4. bubble sort
3. Which of the following sorting algorithm is of divide and conquer type? (1 pt)
   1. Bubble sort
   2. Insertion sort
   3. Merge sort
   4. Selection sort
4. Only problems that are recursively defined can be solved using recursion. True or False? (1 pt)
   1. True
   2. False
5. Recursion and iteration are the same programming approach. True or False? (1 pt)
   1. True
   2. False
6. Which of these is not true about recursion? (1 pt)
   1. The logic behind recursion may be hard to follow
   2. Recursive functions are easy to debug
   3. Recursive calls take up a lot of memory
   4. Programs using recursion take longer time than their non-recursive equivalent
7. Which of the following is not a limitation of binary search algorithm? (1 pt)
   1. must use a sorted array
   2. requirement of sorted array is expensive when a lot of insertion and deletions are needed
   3. there must be a mechanism to access middle element directly
   4. binary search algorithm is not efficient when the data elements more than 1500.