Section 2: Programming

Question 11:
Four hexadecimal digits represent the range from '0000' to 'FFFF'. How many bits are necessary to represent eight hexadecimal digits?

- 16
- 32
- 64
- 128

Question 12:
The series 1+2+3+4+5+6+...+998+999+1000 is equivalent to:

- (1+1000)*500
- 1000*(500+1)
- 1000*500/2
- None of the above

Question 13:
Different sorting algorithms may have very different average time complexities when sorting integer numbers. Using Big-O notation, what is the expected time complexity of, respectively, [1] Bubble sort, [2] QuickSort, [3] Mergesort, [4] Bucket sort (aka Cont sort or Distribution sort)?

- [1] O(n^2), [2] O(n Log(n)), [3] O(log(n)), [4] O(n Log(n))
- [1] O(n^2), [2] O(n Log(n)), [3] O(n Log(n)), [4] O(n)
- [1] O(n Log (n)), [2] O(n Log(n)), [3] O(log(n)), [4] O(n^2)
- None of the above

Question 14:
Djikstra's Algorithm is typically used for:

- finding the overlap between two arbitrary strings.

- finding the elements that intersect between two sets in linear time.
- finding the shortest path in a graph with non-negative edges.
- finding the most likely ancestor between two leaves in a binary tree.

Question 15:

Given a sorted array with N numbers, what assertion is true about Binary Search?

- Binary Search can find an arbitrary element in the array in O(N log(N)) time
- Binary Search can find an arbitrary element in the array in O(N) time
- Binary Search can find an arbitrary element in the array in O(log(N) time
- Linear Search can find an arbitrary element in the array in O(N log(N)) time

Question 16:

Dictionary lookups are often implemented using a Hash Table. What can be said about Hash Tables storing N key-value pairs?

- hash Tables need O(N^2) storage to accomplish fast search times.
- hash Tables cannot be implemented with Linked Lists.
- hash Tables have O(1) average search time.
- hash Tables keep values internally in sorted order.

Question 17:

Divide and Conquer algorithms break down a problem recursively into a larger number of smaller problems that can be solved in a very simple way. Given the following sample programs, what statements can you make about whether they represent a divide and conquer algorithm?

Program A:

```
def programA(inputTree):
  if (inputTree.isLeaf()):
    return inputTree.value()
  else:
    return programA(inputTree.left()) + programA(inputTree.right())
```

Program B:

```
def programB(inputTree):
  sum = 0
  for x in inputTree:
    sum = sum + x
  return sum
```

Program C:

```
def programC(inputTree):
  return helper(0, inputTree)
def helper (sumSoFar, inputTree):
  if (inputTree.isLeaf()):
    return sumSoFar
  else:
    return helper(helper(sumSoFar, inputTree.left()), inputTree.right())
```

- all three compute the same answer and thus are all equivalent
- programs A and C are divide and conquer but program B is iterative
- program A is divide and conquer, program B is iterative, program C is recursive, but not divide and conquer
- none of the programs use divide and conquer methods

Question 18:

A function to traverse a binary search tree data structure "in-order" (aka in-order traversal) can be accomplished by this sequence of operations:

- (1) visit the root, (2) traverse the left subtree, (3) traverse the right subtree.
- (1) visit the root, (2) traverse the right subtree, (3) traverse the left subtree.
- (1) traverse the left subtree, (2) traverse the right subtree, (3) visit the root.
- (1) traverse the left subtree, (2) visit the root, (3) traverse the right subtree.

Question 19:

A Heap is a tree-based data structure frequently used:

- to find the top K largest elements of a set.
- to find the top K smallest elements in a set.
- to find the median value of the elements in a set.
- all of the above.

Question 20:

What does the function Zombie(N) below do?

```
int Zombie(int N){
If (N<1)
   return 1;
Else
return N*Zombie(N-1);
}
```

- calculates the square root of N
-

calculates N factorial

- ○ calculates 2 to the power of N (that is, 2^N)
- ○ calculates the Nth Fibonacci series number

→

UNIVERSITY of WASHINGTON

Data Science Assessment Quiz

This assessment includes 30 multiple-choice questions divided into three sections: Statistics & Linear Algebra, Programming, and Databases & SQL. Applicants must earn a total score of at least 18 out of 30, with a minimum score of 6 out of 10 in each section.

Once you start the assessment, you must complete it within 90 minutes. The timer does not display on the page, so **please make sure to set your own timer** to track how much time has elapsed. After each section, you will be able to see your score, as well as your total score at the end. Please use the same email address on both your assessment and your application.

Applicant Information

Name (Last, First)

Email

→