1. Multiple Choices Questions

Question 1.1 | Two’s Complement

Suppose we use three bits to represent the integers from -4 to 3 using Two’s complement notation. What is the bit representation of -1?

- ☐ `A: 010`
- ☐ `B: 100`
- ☐ `C: 110`
- ☐ `D: 111`

Question 1.2 | Machine Language

Which of the following statements is true about machine language and high-level programming languages?

- ☐ `A: Machine language code requires fewer resources to write and is easier to debug.`
- ☐ `B: High-level programming languages can not be understood by machines directly and require compiling.`
- ☐ `C: Machine language code is portable across different computers.`
- ☐ `D: High-level programming language code runs faster as it hides unnecessary details for programmers.`

Question 1.3 | CPU

When an algorithm runs by a computer with a 6GHz CPU, its time complexity will be ______ running on a computer with a 4GHz CPU.

- ☐ `A: Lower than`
- ☐ `B: Equal to`
- ☐ `C: Higher than`

D: None of the above: depending on other factors of the computer

Question 1.4 | Algorithms

How can we represent an algorithm?

- ☐ A: Natural Language
- ☐ B: Flow chart
- ☐ C: Pseudocode
- ☐ D: All of the above

Question 1.5 | QuickSort

What is the average-case time complexity of QuickSort when pivot selections are not always the smallest or highest?

- ☐ A: O(n)
- ☐ B: O(log(n))
- ☐ C: O(n*log(n))
- ☐ D: O(n^2)

Question 1.6 | Search

Which of the following statements is true about searching algorithms?

- ☐ A: Given an unsorted list, binary search will still work with a time complexity of O(n).
- ☐ B: Given a sorted list, binary search always uses fewer comparisons to find the match than linear search.
- ☐ C: Given an unsorted and small list, linear search can run faster than binary search.
- ☐ D: Given a large list, either linear or binary search will be more efficient than hashing.

Question 1.7 | OOP Programming

What is the expected output of the following code?

```
class Animal:
    def sound(self):
        print("Animal Sound")

class Cat(Animal):
    def sound(self):
        print("Meow")

class Dog(Animal):
    def sound(self):
        print("Bark")

kitty = Dog()
kitty.sound()
```

- [ ] A: Animal Sound
- [ ] B: Meow
- [ ] C: Bark
- [ ] D: None

2. Short Answer Questions

Question 2.1 | When to Sort before Search

You are to look for an item in a list of n items, and you will do the lookup (or search) for m times:

- Just-search: just do m linear searches over the list (i.e., each search scans the entire list)
- Sort-then-search: sort the list first, and then do m searches using binary search.

1. Using Big-O notation, write the complexities of the two approaches. Note: in this case, the time complexity is a function of m and n.

2. Describe when one approach is better than the other.

3. In addition to the just-search and sort-then-search, is there any other approach to complete this task? If yes, please specify the approach and explain its time complexity.

Question 2.2 | Complexity Analysis

1. The following is the code for the selection sort.

```
def selection_sort(arr):
     for i in range(len(arr)):
          min_idx = i
          for j in range(i + 1, len(arr)):
               if arr[j] < arr[min_idx]:
                    min_idx = j
          arr[i], arr[min_idx] = arr[min_idx], arr[i]

     return arr
```

What is the time complexity of the selection sort (assuming arr has N items)? (in Big-O notation)

2. What is the time complexity of the following code? (in Big-O notation)

```
def int_to_binary(x):
    """Assumes x is a nonnegative int
    Returns a binary string representation of x."""

    digits = '01'
    if x == 0:
        return '0'
```

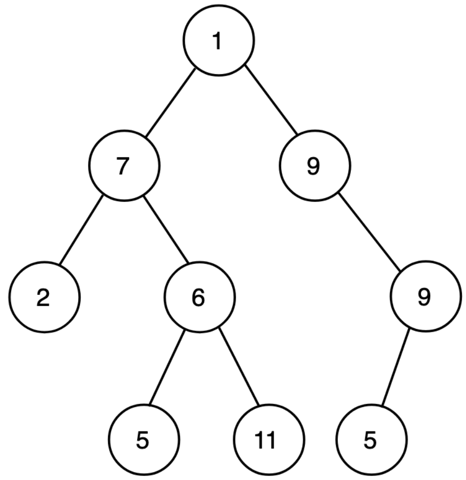
```
    result = ''
    while x > 0:
        result = digits[x%2] + result
        x = x//2
    return result

Output = int_to_binary(N)
```

3. What is the time complexity of the following code? (in Big-O notation)

```
def mystery_function(N):
     sum = []
     i, j = 1, 2
     while i <= N:
          i *= 3
          while j <= N+1:
               sum.append(i+j)
               j *= 2
```

Question 2.3 | Tree Traversal

Here is an example of a binary tree. Write down the different traversals.



Preorder Traversal:____

Inorder Traversal:____

Postorder Traversal:____

3. Programming Questions

Please write your programming solution as clearly as possible.
Try to add comments to explain each code segment.

Question 3.1 | Tribonacci Numbers

In the lecture, we learnt how to use recursion to solve Fibonacci numbers defined as

Fib(0) = 1, Fib(1) = 1, and

```
Fib(n) = Fib(n-1) + Fib(n-2)
```

for all n>=2.

Let us define a new sequence called *Tribonacci numbers* as follows. Tri(0) = 1, Tri(1) = 1, Tri(2) = 1 and

```
Tri(n) = Tri(n-1) + Tri(n-2) + Tri(n-3)
```

for all n>=3.

Please write a code for computing the n-th Tribonacci number using recursion.

```
def Tribonacci_recur(n):
    # write your code here
```

Advance version: write it in dynamic programming. And describe the difference of complexity versus recursion.

```
def Tribonacci_dp(n, mem = {}):
    # write your code here
```

Question 3.2 | Binary Search Returning index

In the lecture, we learnt binary search that returns True or False - whether a given number exists in the list. Now write a function that uses binary search to find the index of the target value in a sorted list. It returns the index of the target value if it is in the list. Otherwise, it returns -1.

```
In [2]: L = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [3]: e = 2

In [4]: bSearch(L, e)
Out[4]: 2

In [5]: e = 9

In [6]: bSearch(L, e)
Out[6]: 9

In [7]: e = 10

In [8]: bSearch(L, e)
Out[8]: -1
```

Below are the arguments for the function input:

- L: the element list to search with;
- e: the target element;
- low=0: index of the first element of L in the given original list, initialized as 0. For example, if the given original list is [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], and the current list L is [5, 6, 7, 8, 9], then, low should be 5(index) since the element 5(value) is the 6th element in the original list.

```
def bSearch(L: list, e: int, low=0):
    # write your code here
```