ENG346 – Data Structures and Algorithms for Artificial Intelligence

Review Questions

Question 1. General understanding.

a. Explain the concept of encapsulation in object-oriented programming. [4 points]

b. Compare and contrast stacks and queues. Provide examples of scenarios where each one is more suitable. [4 points]

c. Explain the concept of list comprehension in Python. Provide example code demonstrating its use. [4 points]

d. Discuss the base case in recursive functions. Why is it necessary? What happens if it is omitted? [4 points]

e. Compare and contrast singly linked lists and doubly linked lists. Discuss the advantages and disadvantages of each. Provide a scenario where using a doubly linked list is beneficial. [4 points]

Question 2. Answer the following questions on recursive algorithms, stacks and queues.

a. Explain how backtracking is used in solving the n-queens solution and maze solution. Provide a high-level overview of a general problem-solving algorithm with backtracking. [5 points]

b. Reverse polish notation, also known as postfix notation, is a mathematical notation in which operators follow their operands. The expression is written without the need for parentheses to indicate the order of operations. An example expression would be "3 4 5 * +." Define a function (in pseudocode) that will evaluate expressions in reverse polish notation. Use necessary ADTs as needed. [5 points]

c. Memoization is a technique to speed up the execution of recursive or computationally expensive functions by *caching the results of function calls* in a suitable data structure and *returning the cached results* from this data structure when the same inputs occur again. Suggest an algorithm, which uses memorization technique to speed up Fibonacci number calculations, i.e. $F_n = F_{n-1} + F_{n-2}$. [5 points]

d. Assuming a queue Q is initially empty, what is the output and the queue content after each of the following operations? Please note: assume a general queue implementation. [5 points]

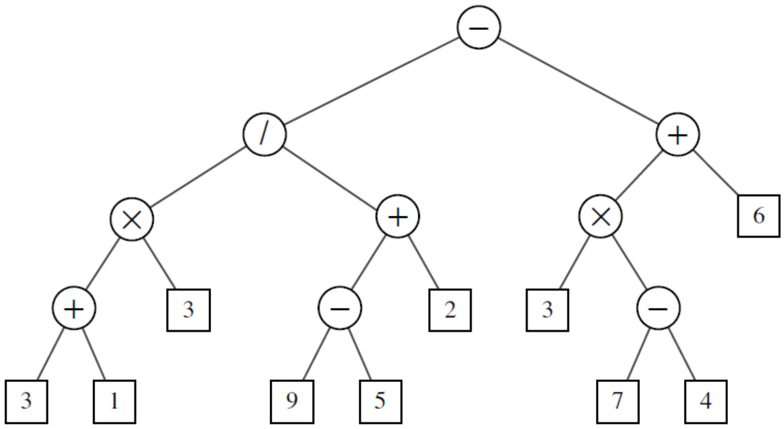
Operation	Return Value	Queue Content
Q.enqueue("Adana")		
Q.enqueue("Ankara")		
Q.first()		
Q.dequeue()		
Q.enqueue("Bilecik")		
Q.enqueue("Bursa")		
Q.first()		
Q.dequeue()		
Q.dequeue()		
Q.dequeue()		

Question 3. Trees

a. What are the similarities and differences between binary tree and binary search tree? [5 points]

b. Define pre-order and post-order traversals on binary trees. [5 points]

c. Given the following binary tree, what is the output of the following code? [4 points]



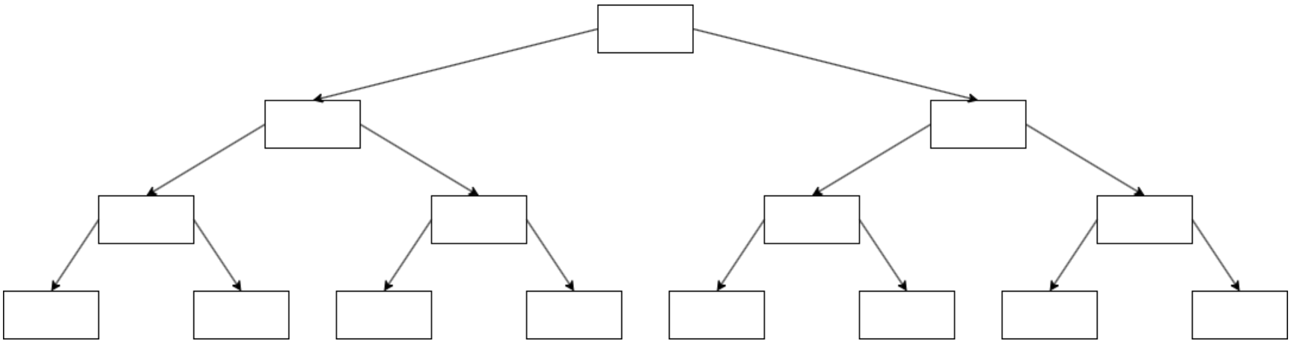
```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

def post_order_traversal(root):
    if root is not None:
        # Traverse the left subtree
        post_order_traversal(root.left)

        # Traverse the right subtree
        post_order_traversal(root.right)

        # Visit the root node
        print(root.value, end=" ")

post_order_traversal(theRoot)
```

Output:

d. Assume that a binary search tree is initially empty.

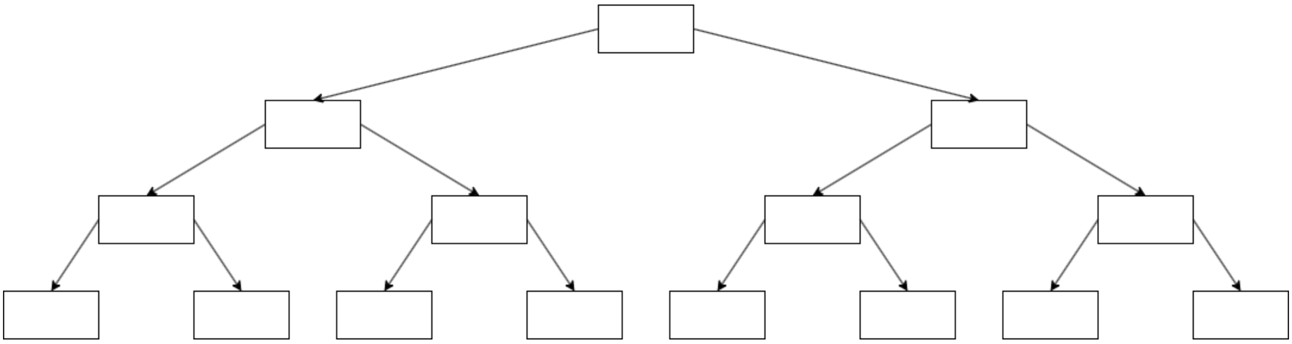
1. Insert the following numbers into this binary search tree. Expand the tree as needed. What is the *height* of the final tree? [3 points]

   [4, 17, 8, 12, 5, 2, 15, 10, 19, 3]



2. Insert the following city names into an empty binary search tree. Expand the tree as needed. What is the *depth* of the "Bursa" node? [3 points]

   ['Istanbul', 'Ankara', 'Izmir', 'Bursa', 'Antalya', 'Adana', 'Konya', 'Gaziantep', 'Mersin', 'Trabzon']



Question 4. Sorting and Searching

a. Given a sequence S of n elements, describe an efficient algorithm for determining whether there are two equal elements in S. What is the runtime performance of your method? [8 points]

b. Define internal and external sorting. Give a real-world example for each case. [4 points]

c. Elaborate on selection of the pivot value for quick-sort algorithm. [4 points]

d. Which sorting algorithms are efficient (i.e. runtime performance) on sorting partially ordered lists? Why? [4 points]

Question 5. Numpy and Pandas

a. Write a function that takes a list of integers as input and produces key statistical metrics for the list, including its sum, length, minimum, maximum, and average. Use Numpy functions as needed. [5 points]

b. Describe Pandas describe() and groupby() functions. [5 points]

c. The following dictionary is given.

```
data = {'Name': ['Alice', 'Bob', 'Charlie', 'Diana'],
       'Age': [25, 30, 22, 27],
       'City': ['New York', 'San Francisco', 'Los Angeles', 'Los Angeles']}
```

1) Create a Pandas DataFrame using this data. [3 points]

2) For the new data frame, what is the output for df.shape? [3 points]

3) For the new data frame, filter the data for Age > 25. Give the code and expected output. [4 points]

Question 6. A *set* is a data structure that stores unique elements of the same type in a sorted order. Write a CustomSet class with the following methods. Add comments as necessary. Assume that the set data is always sorted after add and remove operations. Write down any other assumptions you made.

- A constructor that takes one optional argument, namely *elements*, which can be empty by default. [3 points]
- Add element method, *add(element)*, which will add new *element* to the CustomSet object. [2 points]
- Remove element method, *remove(element)*, which will remove the *element* from the CustomSet object. [2 points]
- Union method, *union(other_set)*, where *other_set* is another CustomSet object. [3 points]
- Intersection method, *intersection(other_set)*, where *other_set* is another CustomSet object. [3 points]
- Difference method, *difference(other_set)*, where *other_set* is another CustomSet object. [4 points]
- Subset method, *subset(other_set)*, where *other_set* is another CustomSet object. The method will return True if $other\_set \subset self$. [3 points]