ENG346 – Data Structures and Algorithms for Artificial Intelligence

Review Questions

Question 1. General understanding.

a. Explain the difference between list and tuple in Python. Provide scenarios where you would choose a list over a tuple and vice versa.

b. Explain the concept of list comprehensions in Python. Provide an example and discuss its advantages over traditional looping.

c. Explain time complexity and space complexity for an algorithm. What is the time complexity of a list traversal function, such as max().

d. Explain the concept of a class and an object in Python. Provide a real-world example.

e. Discuss the purpose and usage of the *__init__()* method in Python classes. Provide an example of a class with an *__init__()* method.

f. How does inheritance promote code reuse in object-oriented programming? Provide an example.

Question 2. Stacks and Queues

a. Implement a Stack Class in Python and explain its basic operations.

b. Assuming a stack S is initially empty, what is the output and the stack content after each of the following operations?

Operation	Return Value	Stack Content
S.push(5)		
S.push(11)		
S.push(7)		
S.top()		
S.pop()		

c. Implement a Queue Class using the Stack Class in the previous question.

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.

Operation	Return Value	Queue Content
Q.enqueue(5)		
Q.enqueue (11)		
Q.enqueue (7)		
Q.first()		
Q.dequeue ()		

Question 3. Recursion

a. Explain the concept of recursion. Provide an example of a recursive function. Elaborate on base case(s) and recursive call(s).

b. Explain the following recursive some_function(). What is the output of this code fragment?

```
def some_function(count):
    if count == 0: # base step
        print('Go!')
    else:
        print(count)
        some_function(count-1) # recursive step
        print(count)

some_function(5)
```

c. Write a short recursive Python function that rearranges a sequence of integer values so that all the even values appear before all the odd values.

Question 4. Linked Lists

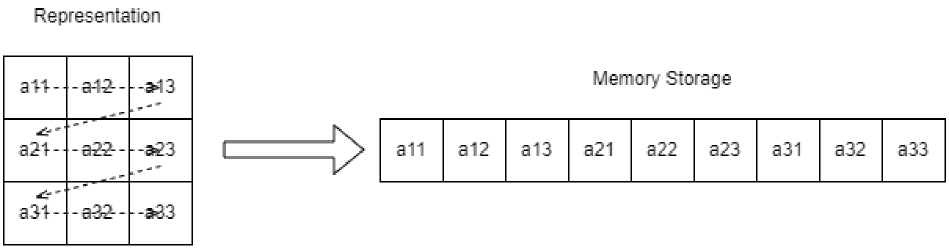
a. Assume that you are implementing Stack ADT using linked lists. Provide *push()* and *pop()* operations in the following code segment.

```
# Node class
class Node:
      def __init__(self, data):
            self.data = data
            self.next = None

# Stack Class
class Stack:

      # head is default NULL
      def __init__(self):
            self.head = None

      # Method to add data to the stack
      def push(self, data):



      # Method to remove data from the stack
      def pop(self):
```

b. Write the function *concatenateLists(L1, L2)* which concatenates two singly linked lists and return a singly linked list.

Question 5. Answer the following questions regarding multi-dimensional arrays.

Multi-dimensional arrays are commonly stored as row-major order in computer memory. In row-major order, the elements of each row are stored contiguously in memory, and the rows themselves are stored in a sequential manner. Visualization of this technique is given in the following figure:



Design and code a Python *class*, named *TwoDimClass*, which uses single list of elements, and implements row-major order to access individual list elements. Your class shall include the following methods:

- A constructor, which takes two arguments, namely *M*, number of rows, and *N*, number of columns.
- A getter function, *getByIndex(m,n)*, where m is the row number, and n is the column number.
- A setter function, *setByIndex(m,n)*, where m is the row number, and n is the column number.
- A row getter function, getByRow(m), where m is the row number.
- A column getter function, getByColumn(n), where n is the column number.

Question 6. Answer the following questions regarding Hailstone squences.

The Hailstone sequence is a mathematical sequence defined by the following rules:

1. Start with any positive integer n
2. If n is even, the next number in the sequence will be n / 2
3. If n is odd, the next number in the sequence will be 3n + 1
4. Continue until the sequence reaches 1.

More formally, *HailStone(n)* returns a list L such that

$$L[i] \quad \begin{cases} n & \text{if } i = 0 \\ L[i-1]/2 & \text{if } L[i-1] \text{ is even} \\ 3 * L[i-1] + 1 & \text{if } L[i-1] \text{ is odd} \end{cases}$$

Example: HailStone(11) returns [11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]

a. What is the return value for HailStone(17)?

b. Write HailStone(n) function using loops.

c. Write HailStoneRecursive(n) function using recursion.

d. Write HailStoneLen(n) function such that the function returns the length of the HailStone(n) sequence. Please note: *len(HailStone(n))* is not the answer!