MULTIMEDIA UNIVERSITY OF KENYA

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

UNIVERSITY EXAMINATIONS 2023/2024

SECOND YEAR FIRST SEMESTER EXAMINATION FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

UNIT CODE: CCS 2214 UNIT NAME: DATA STRUCTURES AND ALGORITHMS

DATE: DECEMBER 2023 TIME: 2 HOURS

INSTRUCTIONS:

ANSWER YOUR QUESTIONS IN ANSWER BOOKLET PROVIDED.

ANSWER QUESTION ONE [COMPULSORY] AND ANY OTHER TWO QUESTIONS.

QUESTION ONE (THIRTY MARKS)

a) Explain in a sentence or two why understanding data structures and algorithms is crucial in computer science. (4 Marks)

b) Provide an example of a real-world situation where a big-tech company has chosen the right data structure to significantly improve the performance of an application. Mention the particular data structure and algorithm that was used in that case. (4 Marks)

c) Explain the concept of "recursion depth" and how it relates to the efficiency and potential issues in recursive algorithms. (4 Marks)

d) You are given two algorithms: Algorithm A with a time complexity of O(2^n) and Algorithm B with a time complexity of O(n^2). Explain which algorithm is more efficient and why, especially as the input size grows. Sketch a graph to show how each of the time functions will as the input size grows. (5 Marks)

e) Write the algorithm (pseudocode) for deleting a node in a doubly linked list. (6 Marks)

f) Write a Python function to remove duplicates from an unsorted linked list. You are given the head of the linked list, and your task is to modify the list in-place to ensure that no two nodes have the same value. The order of the remaining elements should be preserved. You need to implement the remove_duplicates function that takes the head of the linked list as input and returns the modified head of the linked list.
Ensure that your solution works for various test cases and has a time complexity of O(n) where 'n' is the number of nodes in the linked list. (7 Marks)
Example:

```
Input: 1 -> 3 -> 2 -> 1 -> 4 -> 3 -> 5
Output: 1 -> 3 -> 2 -> 4 -> 5
```

QUESTION TWO (TWENTY MARKS)

a) What is the primary difference between primitive and non-primitive data structures? Provide an example of each. (2 Marks)
b) When might you choose to use an iterative approach over a recursive one, and vice versa? Provide examples to illustrate your answer. (4 Marks)
c) List at least three singly linked list operations and give and explain their time complexity. (6 Marks)
d) Using the "3 Steps" procedure we learnt. Explain how and write a recursive function to find the greatest common divisor (GCD) of two positive integers using the Euclidean algorithm. (8 Marks)

QUESTION THREE (TWENTY MARKS)

a) If you were tasked with selecting an algorithm for a real-world application where speed is of utmost importance, would you prioritize an algorithm with a lower or higher Big O time complexity? Justify your choice and provide an example of an algorithm suitable for the scenario you describe. (4 Marks)

b) When implementing a stack data structure, you have the option to use either a List or a Linked List. List their respective strengths and weaknesses in this context. (6 Marks)

c) You are tasked with implementing a binary search tree (BST) in Python. Create a class BinarySearchTree with the following operations:
insert(value): Insert a value into the binary search tree while maintaining the BST property.
search(value): Search for a given value in the tree. Return True if the value is found, otherwise return False.
delete(value): Delete a node with the given value from the tree while preserving the BST structure.
Write the Python class BinarySearchTree and provide the code for the three operations mentioned above. Include any helper functions or attributes you consider necessary.

You can assume the following class definition to get started:

```python
class Node:
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

class BinarySearchTree:
    def __init__(self):
        # Initialize an empty binary search tree
        pass

    # Your code for insert, search, and delete operations goes here

# Example usage:
bst = BinarySearchTree()
bst.insert(10)
bst.insert(5)
bst.insert(15)
print(bst.search(10))  # Should print True
bst.delete(10)
print(bst.search(10))  # Should print False
```

Please provide the code for the BinarySearchTree class with the requested operations, and make sure it adheres to the principles of a binary search tree. (10 Marks)

QUESTION FOUR (TWENTY MARKS)

a) What is a greedy algorithm? (2 Marks)
b) Briefly explain at least four sorting algorithms that you are familiar with. (8 Marks)
c) You are tasked with implementing a Python class for an undirected graph and solving a specific problem using this graph representation. Create a class Graph and write a method find_shortest_path that finds the shortest path between two nodes in the graph.

Your Graph class should have the following methods:

__init__(self): Initializes an empty graph.
add_edge(self, u, v): Adds an undirected edge between nodes u and v in the graph.
find_shortest_path(self, start, end): Finds and returns the shortest path between the start and end nodes as a list of nodes. If there is no path between the nodes, return an empty list.

Consider using a breadth-first search (BFS) algorithm to find the shortest path.

You will be graded based on the following criteria:

- Correctness and functionality of the Graph class.
- Proper implementation of the find_shortest_path method.
- Handling of cases where there is no path between the given nodes.
- Code clarity and readability.

Here is an example of how your class should be used:

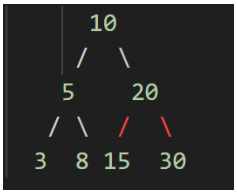
```
# Example usage:
graph = Graph()
graph.add_edge('A', 'B')
graph.add_edge('A', 'C')
graph.add_edge('B', 'D')
graph.add_edge('D', 'E')
graph.add_edge('C', 'F')

print(graph.find_shortest_path('A', 'E'))  # Should print ['A', 'B', 'D', 'E']
print(graph.find_shortest_path('A', 'F'))  # Should print ['A', 'C', 'F']
print(graph.find_shortest_path('B', 'F'))  # Should print []
```

Please provide the Python class Graph and the find_shortest_path method, adhering to the given requirements. (10 Marks)

QUESTION FIVE (TWENTY MARKS)

a) Compare and contrast between:
   i. Weighted and Unweighted graph (2 Marks)
   ii. Cyclic and Acyclic graph (2 Marks)

b) Consider the following Binary Search Tree (BST):



Identify and perform the three depth first search traversals on the given BST. Provide the sequence of visited nodes for each of the three traversals. (6 Marks)

c) An animal shelter, which holds only dogs and cats, operates on a strictly "first in, first out" basis. People must adopt either the "oldest" (based on arrival time) of all animals at the shelter, or they can select whether they would prefer a dog or a cat (and will receive the oldest animal of that type). They cannot select which specific animal they would like. Create the data structures to maintain this system and implement operations such as enqueue, dequeueAny, dequeueDog, and dequeueCat. (10 Marks)