Data Structures: Exercises

Prof. Dr. Thomas Schuster

Winter Semester 2024/25

1. (10 points) **Basic Stack Operations**

   Given the following Python class implementing a Stack, write the code to perform the following operations:

   1. Push ‘10‘, ‘20‘, and ‘30‘ onto the stack.
   2. Pop an element from the stack and display the remaining elements.
   3. Check if the stack is empty.

   Stack Class:

```python
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, item):
        self.stack.append(item)

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()

    def is_empty(self):
        return len(self.stack) == 0
```

   Answer:

2. (10 points) **Reverse a Stack**

   Implement a function `reverse_stack()` that reverses a stack using recursion. You can use the following `Stack` class:

```python
class Stack:
    def __init__(self):
        self.stack = []

    def push(self, item):
        self.stack.append(item)

    def pop(self):
        if not self.is_empty():
            return self.stack.pop()

    def is_empty(self):
        return len(self.stack) == 0
```

   Answer:

3. (10 points) **Debugging a Queue**

   Consider the following buggy Python code that implements a `Queue`. Use the VS Code debugger to identify and fix the bug that prevents the `dequeue` operation from working as expected.

```python
class Queue:
    def __init__(self):
        self.queue = []

    def enqueue(self, item):
        self.queue.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.queue.pop(0)  # Bug here, fix required

    def is_empty(self):
        return len(self.queue) == 0
```

   Use VS Code debugging tools like breakpoints, stepping through the code, and checking variable values.

   Answer:

4. (10 points) **Linked List Operations**

   You are given a Python implementation of a Singly Linked List. Implement a method to remove the node with a given value from the linked list.

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

class LinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node
```

   Write the code to remove a node with a given value.

   Answer:

5. (20 points) **Group Assignment: Explore and Present**

   Objective: In this group assignment, your task is to research a data structure that we have not yet discussed in the course. You will work together to understand its properties, implement it in Python, and create a short presentation that explains how it works and where it can be applied.

   Instructions:

   (a) **Research**

   - Choose a data structure that we have not covered in class.
   - Answer the following questions:
     - What is the basic concept behind this data structure?
     - How do the operations of insertion, searching, and deletion work?
     - What are real-world use cases where this data structure is typically applied?

   (b) **Implementation in Python**

   - Implement, extend or utilize (e.g., writing tests) the data structure in Python, ensuring that your implementation includes the core operations, such as insertion, searching, and (if applicable) deletion.
   - Add comments to your code to make it easy to understand for someone else.

   (c) **Prepare a Presentation Slides**

   - Create presentation slides that includes:
     - A brief explanation of the data structure (include diagrams if helpful).
     - A description of its key operations (insertion, searching, deletion).
     - A code snippet from your Python implementation that demonstrates an important operation.
     - Examples that show how this data structure might be practically used.
     - A short conclusion on why this data structure is useful and when it might be preferred over others.

   (d) **Submission:** Each group must upload their final slide as a PDF file to Moodle before our next class in November. Make sure file includes your group and team member names.

   Note: You are encouraged to divide the workload among your group members. One person might focus on research, another on implementation, and another on the presentation slide. However, make sure that every group member understands the entire process.