

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING		
Program Name: M. Tech/MCA/MSC		Assignment Type: Lab		
Course Coordinator Name		Venkataramana Veeramsetty		
Course Code		Course Title	AI Assisted Problem Solving Using Python	
Year/Sem	II/I	Regulation	R24	
Date and Day of Assignment	Week 5- Monday	Time(s)		
Duration	2 Hours	Applicable to Batches		
Assignment Number: 11.3(Present assignment number)/ 24 (Total number of assignments)				

Q.No.	Question	Expected Time to complete
1	<p>Lab 11 – Data Structures with AI: Implementing Fundamental Structures</p> <p>Lab Objectives</p> <ul style="list-style-type: none"> To implement fundamental data structures with the assistance of AI tools. To understand how AI suggests different implementations and optimizations. To analyze the readability, correctness, and performance of AI-generated code. To reinforce problem-solving skills using AI-powered coding assistance. <p>Learning Outcomes</p> <p>After completing this lab, students will be able to:</p> <ol style="list-style-type: none"> Implement stack, queue, and linked list using Python with AI support. Use AI tools to optimize and refactor basic data structure operations. Compare multiple AI-suggested implementations for the same structure. Apply AI assistance to generate test cases for verifying data structure behavior. Demonstrate understanding of trade-offs in AI-generated solutions. <p>Task Description #1 – Stack class implementation</p> <p>Task: Ask AI to implement a stack class with push(), pop(), peek() and is_empty() methods</p> <p>Task Description #2 – Queue Implementation</p>	Week 5 - Monday

	<p>Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is_empty().</p> <p>Task Description #3 – Linked List Implementation</p> <p>Task: Ask AI to create a singly linked list with insert_at_end(), insert_at_beginning(), and display().</p> <p>Task Description #4 – Binary Search Tree (BST)</p> <p>Task: Ask AI to generate a simple BST with insert() and inorder_traversal().</p>	
--	---	--

Task Description #1 – Stack class implementation

Task: Ask AI to implement a stack class with push(), pop(), peek() and is_empty() methods

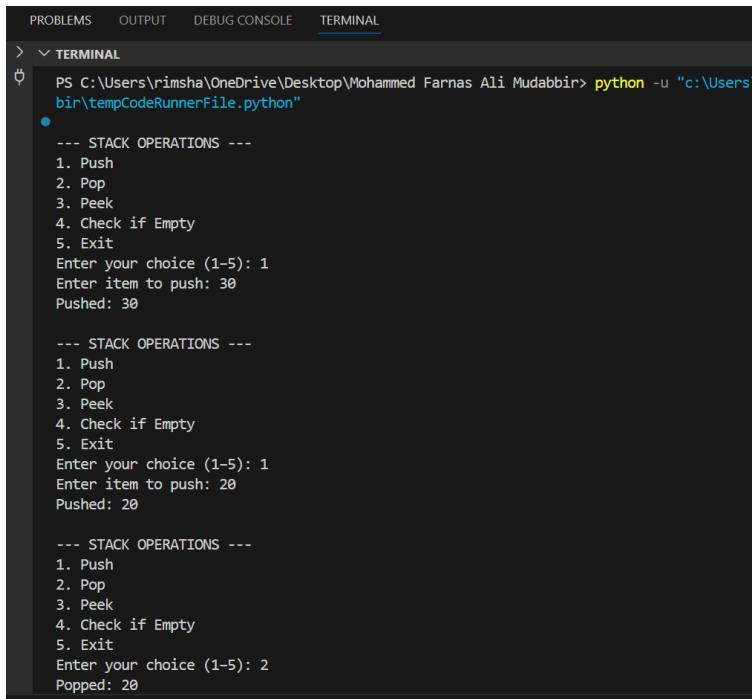
PROMPT: Write a Python program that takes user input to operate a Stack class. Implement methods: push(), pop(), peek(), and is_empty(). Let the user enter values to push and choose operations.

```
 TASK 2.py X  TASK 3.py  TASK 4.py  ●  TASK 5.py  TASK 6.py  TASK 1.py Mohammed Farnas Ali Mudabbir • LAB
Mohammed Farnas Ali Mudabbir > LAB 11 > ✎ TASK 1.py > ...
1  class Stack:
2      """
3          A simple Stack implementation using Python list.
4          Supports push, pop, peek, and is_empty operations.
5      """
6
7      def __init__(self):
8          """Initialize an empty stack."""
9          self.items = []
10
11     def push(self, item):
12         """Push an item onto the stack."""
13         self.items.append(item)
14
15     def pop(self):
16         """Pop and return the top item of the stack."""
17         if self.is_empty():
18             return "Stack is empty. Cannot pop."
19         return self.items.pop()
20
21     def peek(self):
22         """Return the top item without removing it."""
23         if self.is_empty():
24             return "Stack is empty. Nothing to peek."
25         return self.items[-1]
26
27     def is_empty(self):
28         """Check if the stack is empty."""
29         return len(self.items) == 0
```

```
Mohammed Farnas Ali Mudabbir > LAB 11 > ✎ TASK 1.py > ...
32  # ----- MENU DRIVEN USER INPUT -----
33
34  stack = Stack()
35
36  while True:
37      print("\n--- STACK OPERATIONS ---")
38      print("1. Push")
39      print("2. Pop")
40      print("3. Peek")
41      print("4. Check if Empty")
42      print("5. Exit")
43
44      choice = input("Enter your choice (1-5): ")
45
46      if choice == "1":
47          item = input("Enter item to push: ")
48          stack.push(item)
49          print("Pushed:", item)
50
51      elif choice == "2":
52          print("Popped:", stack.pop())
53
54      elif choice == "3":
55          print("Top Element:", stack.peek())
56
57      elif choice == "4":
58          print("Is Stack Empty?:", stack.is_empty())
```

```
57     elif choice == "4":  
58         print("Is Stack Empty?", stack.is_empty())  
59  
60     elif choice == "5":  
61         print("Exiting program...")  
62         break  
63  
64     else:  
65         print("Invalid choice! Please enter between 1-5.")  
66
```

PRACTICAL OUTPUT:



The screenshot shows a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
> ^ TERMINAL  
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\bir\tempCodeRunnerFile.python"  
● --- STACK OPERATIONS ---  
1. Push  
2. Pop  
3. Peek  
4. Check if Empty  
5. Exit  
Enter your choice (1-5): 1  
Enter item to push: 30  
Pushed: 30  
  
--- STACK OPERATIONS ---  
1. Push  
2. Pop  
3. Peek  
4. Check if Empty  
5. Exit  
Enter your choice (1-5): 1  
Enter item to push: 20  
Pushed: 20  
  
--- STACK OPERATIONS ---  
1. Push  
2. Pop  
3. Peek  
4. Check if Empty  
5. Exit  
Enter your choice (1-5): 2  
Popped: 20
```

```
> ▾ TERMINAL
⚡ PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> pyt
bir\tempCodeRunnerFile.python"
--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 10
Invalid choice! Please enter between 1-5.

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 3
Top Element: 30

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 4
Is Stack Empty?: False

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 4
Is Stack Empty?: False

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 5
Exiting program...
⚡ PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> []
```

Task Description #2 – Queue Implementation

Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is_empty().

PROMPT: Write a Python program that takes user input to create a Queue class with enqueue(), dequeue(), and is_empty() methods. Then let the user add and remove items from the queue.

```
Mohammed Farnas Ali Mudabbir > LAB 11 > TASK 2.py > ...
1  class Queue:
2      def __init__(self):
3          self.items = []
4
5      def enqueue(self, item):
6          self.items.append(item)
7
8      def dequeue(self):
9          if self.is_empty():
10              return "Queue is empty"
11          return self.items.pop(0)
12
13      def is_empty(self):
14          return len(self.items) == 0
15
16
17  # ---- USER INPUT ----
18 q = Queue()
19
20 # Enqueue two items from user
21 q.enqueue(input("Enter first item: "))
22 q.enqueue(input("Enter second item: "))
23
24 print("Dequeue:", q.dequeue())
25 print("Is queue empty?", q.is_empty())
26 print("Dequeue:", q.dequeue())
27 print("Is queue empty?", q.is_empty())
28 |
```

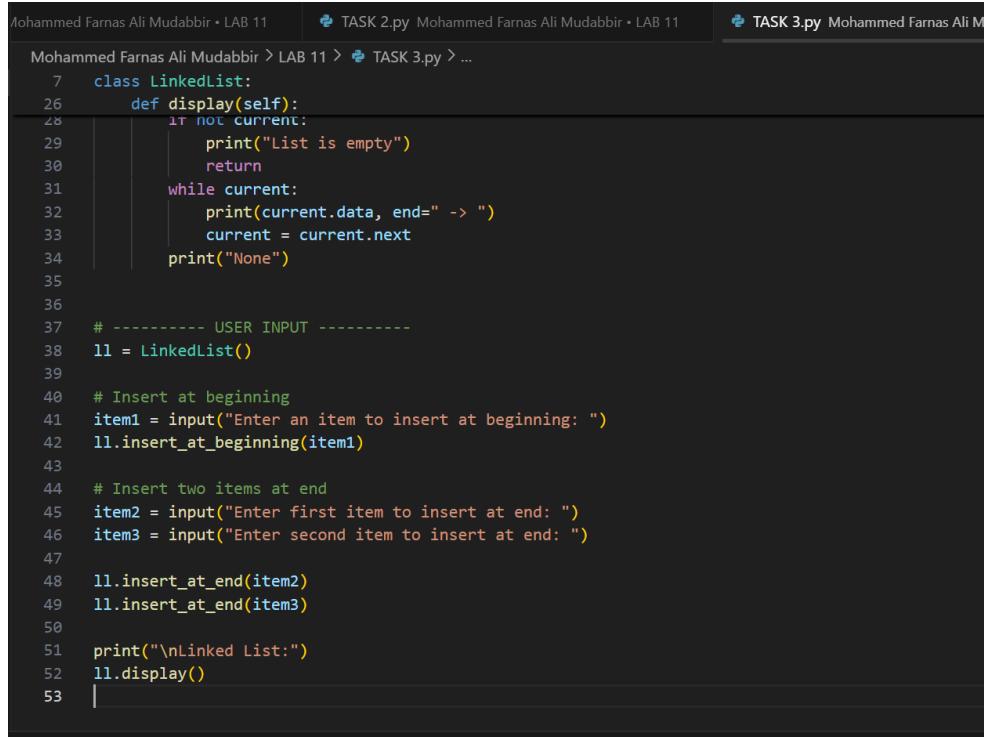
PRACTICAL OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG  TERMINAL
> ▾ TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
Enter first item: A
Enter second item: B
Dequeue: A
Is queue empty? False
Dequeue: B
Is queue empty? True
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```

Task Description #3 – Linked List Implementation

Task: Ask AI to create a singly linked list with insert_at_end(), insert_at_beginning(), and display().

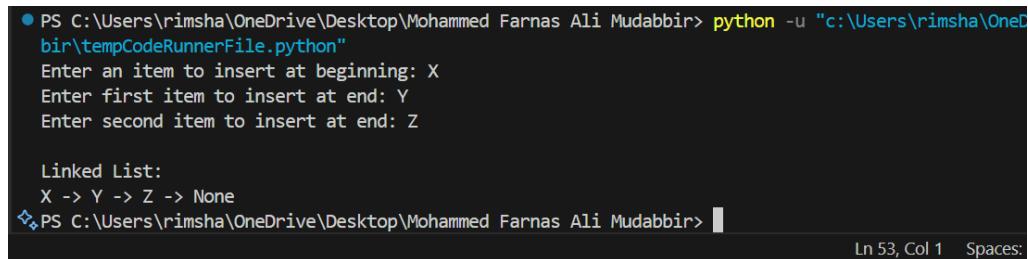
PROMPT: Create a Python program for a singly linked list with insert_at_beginning(), insert_at_end(), and display() methods. Take user input to insert and view the list.



The screenshot shows a code editor with two tabs: "TASK 2.py" and "TASK 3.py". The "TASK 3.py" tab is active, displaying the following Python code:

```
1 Mohammed Farnas Ali Mudabbir • LAB 11
2 TASK 2.py Mohammed Farnas Ali Mudabbir • LAB 11
3 TASK 3.py Mohammed Farnas Ali Mudabbir • LAB 11
4
5 Mohammed Farnas Ali Mudabbir > LAB 11 > TASK 3.py > ...
6
7 class LinkedList:
8     def display(self):
9         if not current:
10             print("List is empty")
11             return
12         while current:
13             print(current.data, end=" -> ")
14             current = current.next
15         print("None")
16
17
18 # ----- USER INPUT -----
19 ll = LinkedList()
20
21 # Insert at beginning
22 item1 = input("Enter an item to insert at beginning: ")
23 ll.insert_at_beginning(item1)
24
25 # Insert two items at end
26 item2 = input("Enter first item to insert at end: ")
27 item3 = input("Enter second item to insert at end: ")
28
29 ll.insert_at_end(item2)
30 ll.insert_at_end(item3)
31
32 print("\nLinked List:")
33 ll.display()
```

PRACTICAL OUTPUT:



The screenshot shows a terminal window with the following output:

```
● PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.python"
Enter an item to insert at beginning: X
Enter first item to insert at end: Y
Enter second item to insert at end: Z

Linked List:
X -> Y -> Z -> None
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```

Task Description #4 – Binary Search Tree (BST)

Task: Ask AI to generate a simple BST with insert() and inorder_traversal().

PROMPT: Write a Python program to implement a Binary Search Tree with insert() and inorder_traversal(), and allow user input to add nodes.

```
Mohammed Farnas Ali Mudabbir • LAB 11 | TASK 3.py Mohammed Farnas Ali Mudabbir • LAB 11
Mohammed Farnas Ali Mudabbir > LAB 11 > TASK 4.py > ...

1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.left = None
5          self.right = None
6
7
8  class BST:
9      def __init__(self):
10         self.root = None
11
12     def insert(self, data):
13         self.root = self._insert_recursive(self.root, data)
14
15     def _insert_recursive(self, node, data):
16         if node is None:
17             return Node(data)
18         if data < node.data:
19             node.left = self._insert_recursive(node.left, data)
20         else:
21             node.right = self._insert_recursive(node.right, data)
22         return node
23
24     def inorder_traversal(self):
25         print("\nInorder Traversal:", end=" ")
26         self._inorder(self.root)
27         print()
28
29         self._inorder(self.root)
30         print()
31
32     def _inorder(self, node):
33         if node:
34             self._inorder(node.left)
35             print(node.data, end=" ")
36             self._inorder(node.right)
37
38
39     # ----- USER INPUT -----
40     bst = BST()
41
42     n = int(input("How many values do you want to insert? "))
43
44     for _ in range(n):
45         value = int(input("Enter value: "))
46         bst.insert(value)
47
48     bst.inorder_traversal()
```

PRACTICAL OUTPUT:

```
TERMINAL

PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir> python -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\tempCodeRunnerFile.py"
● How many values do you want to insert? 6
Enter value: 40
Enter value: 80
Enter value: 20
Enter value: 50
Enter value: 70
Enter value: 10

Inorder Traversal: 10 20 40 50 70 80
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir>
```