

Assignment On

‘ADVANCED DATA STRUCTURES AND ALGORITHMS’ (Assignment-3)

Submitted by:
K. Shiva Kiran
2503B05121(M.Tech).
Submitted to:
Dr. Rahul Mishra.

Question: Create an Undo-Redo System Using Two Stacks. Implement Push, Pop, and Display Operations.

Code :

```
class UndoRedoSystem:  
  
    def __init__(self):  
        self.undo_stack = []  
        self.redo_stack = []  
  
        # Push a new action    def  
        push_action(self, action):  
            self.undo_stack.append(action)    self.redo_stack.clear() # Clear  
redo history when a new action happens    print(f"Action added:  
{action}")  
  
        # Undo operation    def  
        undo(self):    if not  
self.undo_stack:  
            print("Nothing to undo.")  
return
```

```
action = self.undo_stack.pop()
self.redo_stack.append(action)

print(f"Undo: {action}")

# Redo operation      def
redo(self):           if not
self.redo_stack:
print("Nothing    to    redo.")

return

action = self.redo_stack.pop()
self.undo_stack.append(action)
print(f'Redo: {action}')

# Display both stacks
def display(self):
    print("\n---    Undo-Redo    Status    ---")
    print("Undo    Stack:", self.undo_stack)
    print("Redo Stack:", self.redo_stack)    print("-"
-----")
# -----
# Example Usage
# -----
system = UndoRedoSystem()

system.push_action("Type      A")
system.push_action("Type      B")
system.push_action("Type C")
```

```
system.display()
```

```
system.undo()
```

```
system.undo()
```

```
system.display()
```

```
system.redo()
```

```
system.display()
```

Output:

```
Users/HP/Desktop/M.TECH/ADSA/Assignment 3/task 3.py"
Action added: Type A
Action added: Type B
Action added: Type C

--- Undo-Redo Status ---
Undo Stack: ['Type A', 'Type B', 'Type C']
Redo Stack: []
-----
Undo: Type C
Undo: Type B

--- Undo-Redo Status ---
Undo Stack: ['Type A']
Redo Stack: ['Type C', 'Type B']
-----
Redo: Type B

--- Undo-Redo Status ---
Undo Stack: ['Type A', 'Type B']
Redo Stack: ['Type C']
```