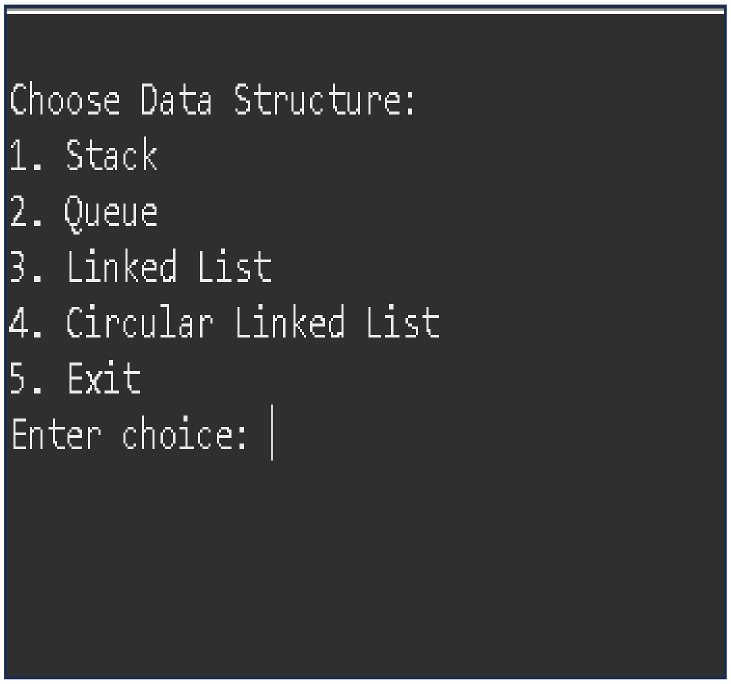
This is the first appearance when we run the program, you need to choose what data structure you will be using and input its corresponding number.

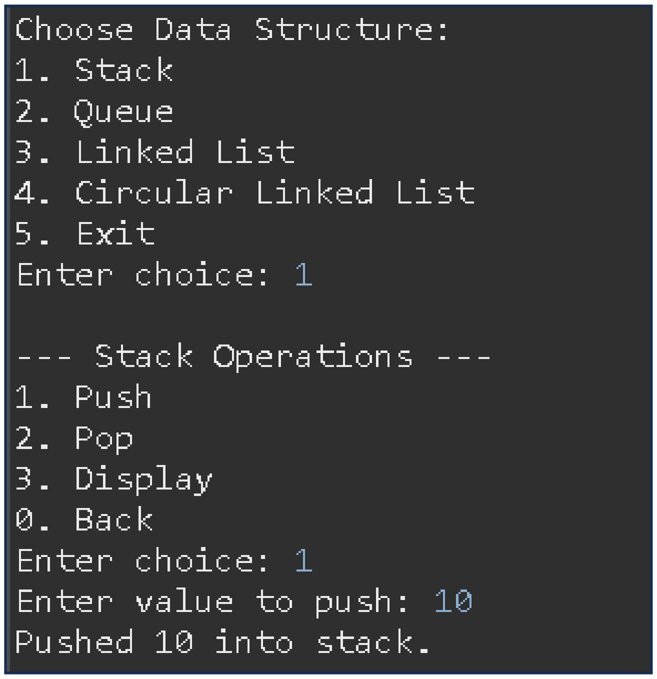


```
Choose Data Structure:
1. Stack
2. Queue
3. Linked List
4. Circular Linked List
5. Exit
Enter choice: 
```

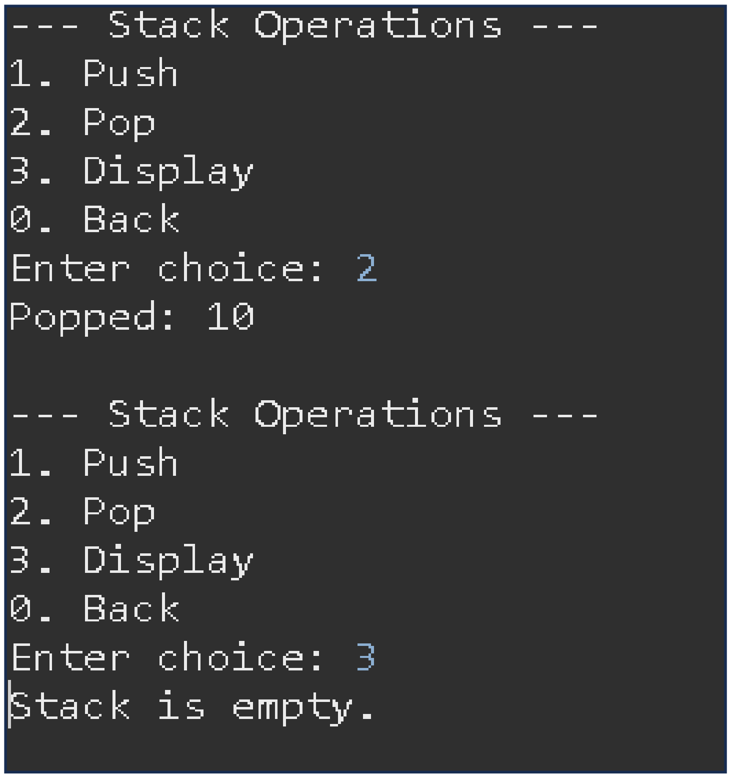
1. Stack (MyStack class)

- **Push** → Adds a new element on top of the stack.
- **Pop** → Removes the top element from the stack.
- **Display** → Shows all elements in the stack (from top to bottom).

It follows LIFO (Last In, First Out) principle.



```
Choose Data Structure:
1. Stack
2. Queue
3. Linked List
4. Circular Linked List
5. Exit
Enter choice: 1

--- Stack Operations ---
1. Push
2. Pop
3. Display
0. Back
Enter choice: 1
Enter value to push: 10
Pushed 10 into stack.
```

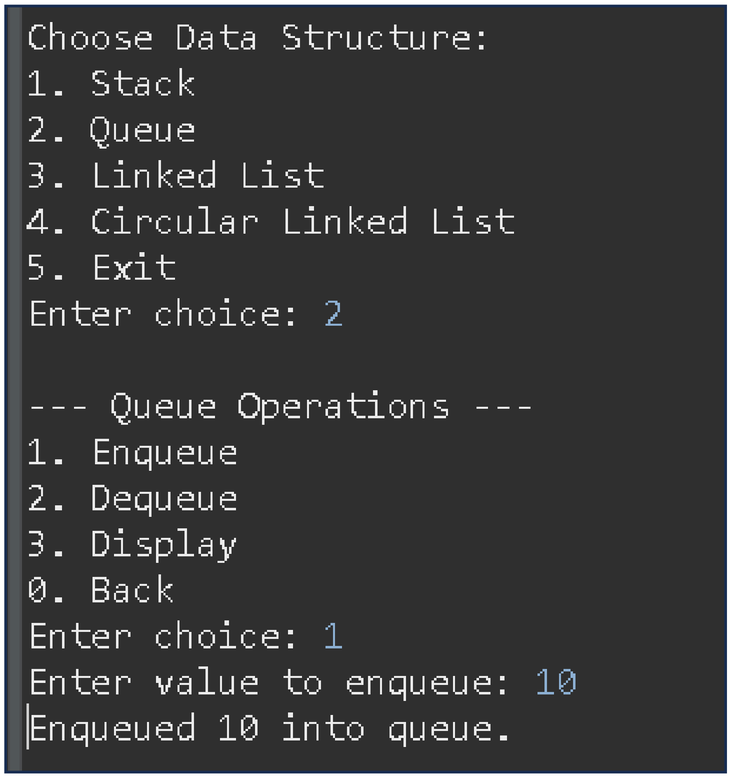


```
--- Stack Operations ---
1. Push
2. Pop
3. Display
0. Back
Enter choice: 2
Popped: 10

--- Stack Operations ---
1. Push
2. Pop
3. Display
0. Back
Enter choice: 3
Stack is empty.
```

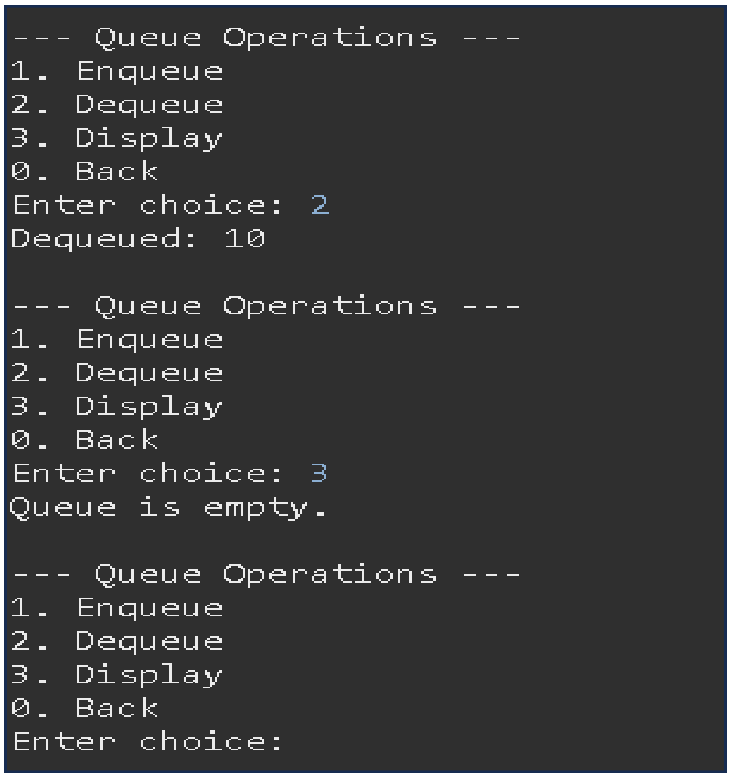
2. Queue (MyQueue class)

- **Enqueue** → Adds a new element at the end of the queue.
- **Dequeue** → Removes the element at the front of the queue.
- **Display** → Shows all elements in the queue in order.

It follows FIFO (First In, First Out) principle.



```
Choose Data Structure:
1. Stack
2. Queue
3. Linked List
4. Circular Linked List
5. Exit
Enter choice: 2

--- Queue Operations ---
1. Enqueue
2. Dequeue
3. Display
0. Back
Enter choice: 1
Enter value to enqueue: 10
Enqueued 10 into queue.
```

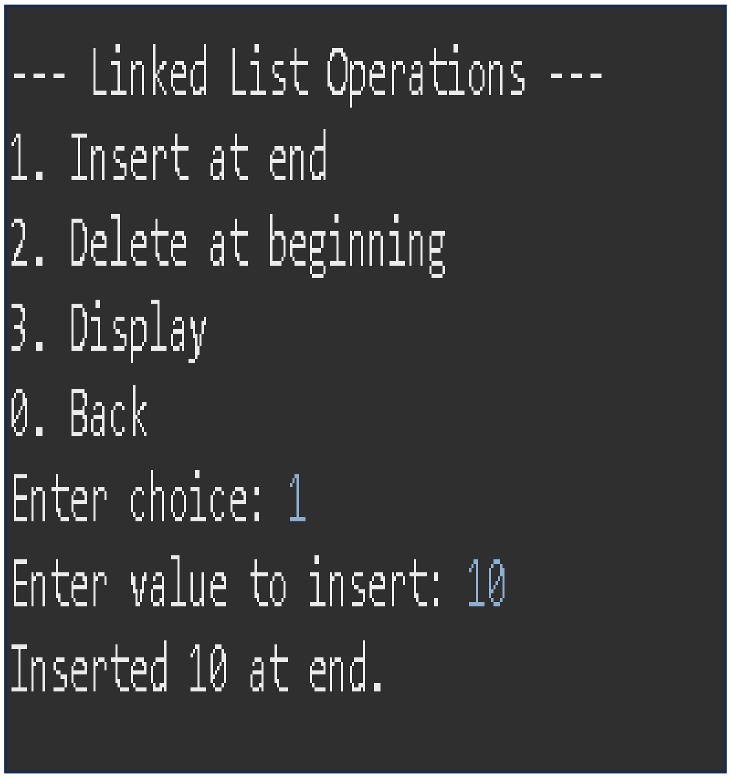


```
--- Queue Operations ---
1. Enqueue
2. Dequeue
3. Display
0. Back
Enter choice: 2
Dequeued: 10

--- Queue Operations ---
1. Enqueue
2. Dequeue
3. Display
0. Back
Enter choice: 3
Queue is empty.

--- Queue Operations ---
1. Enqueue
2. Dequeue
3. Display
0. Back
Enter choice:
```

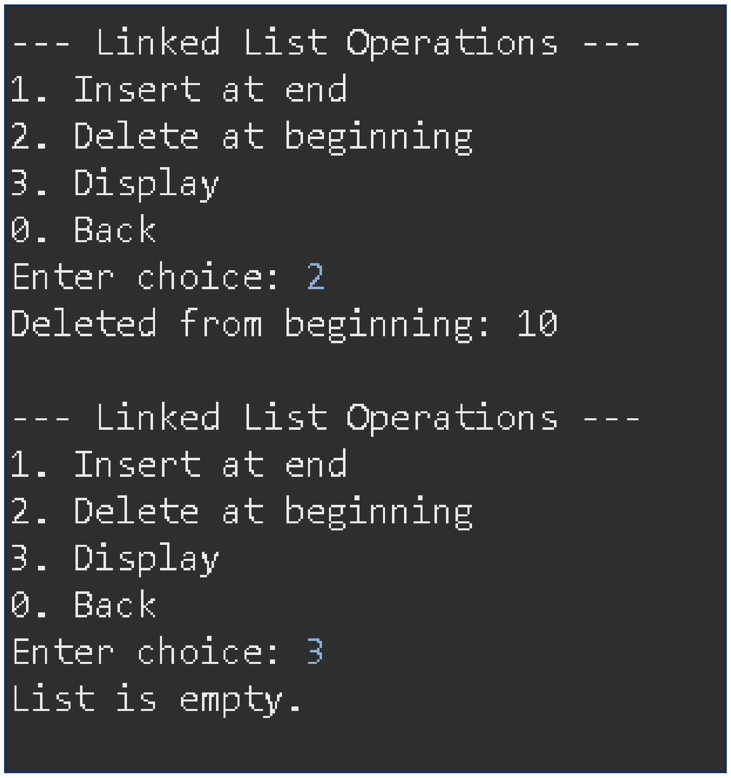
3. Linked List (MyLinkedList class)

- **Insert at End** → Appends a new element to the end of the list.
- **Delete at Beginning** → Removes the first element of the list.
- **Display** → Prints all nodes in the list.

It uses a simple linear list structure.



```
--- Linked List Operations ---
1. Insert at end
2. Delete at beginning
3. Display
0. Back
Enter choice: 1
Enter value to insert: 10
Inserted 10 at end.
```

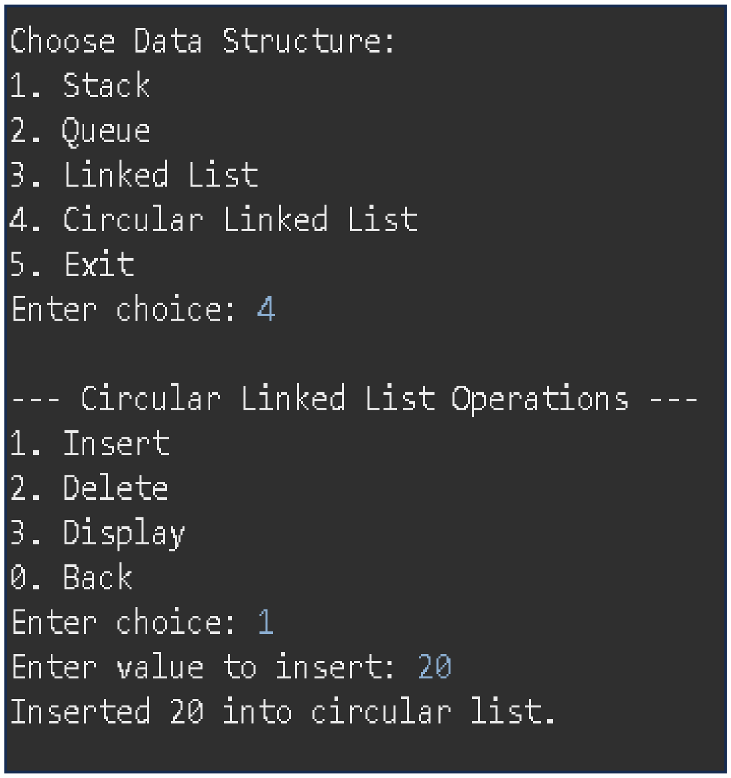


```
--- Linked List Operations ---
1. Insert at end
2. Delete at beginning
3. Display
0. Back
Enter choice: 2
Deleted from beginning: 10

--- Linked List Operations ---
1. Insert at end
2. Delete at beginning
3. Display
0. Back
Enter choice: 3
List is empty.
```

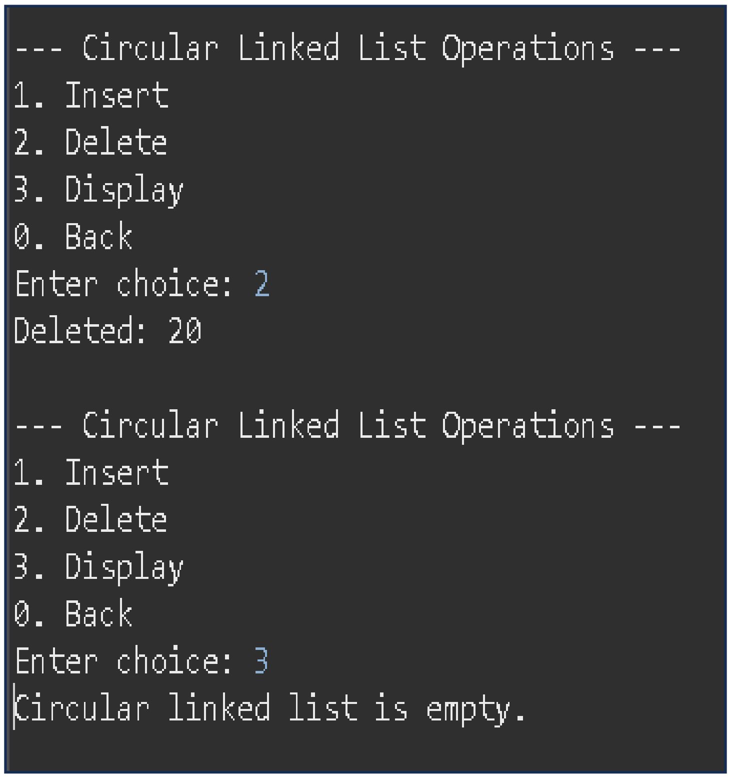
4. Circular Linked List (MyCircularLinkedList)

- **Insert** → Adds a new element at the end (tail links back to head).
- **Delete** → Removes the first element (updates head, keeps circular link).
- **Display** → Traverses and prints elements until it loops back to head.

The last node points back to the first node, forming a circle.

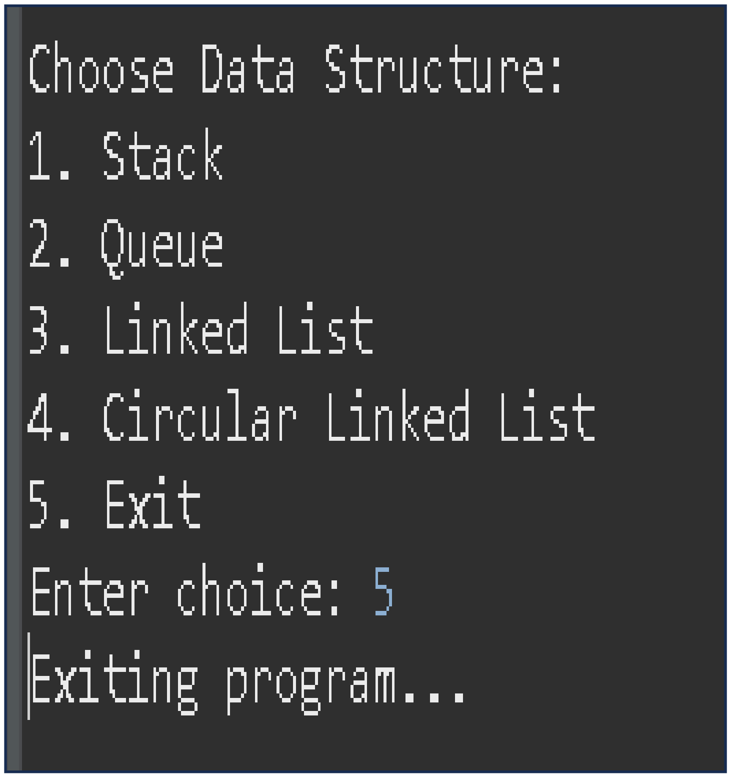


```
Choose Data Structure:
1. Stack
2. Queue
3. Linked List
4. Circular Linked List
5. Exit
Enter choice: 4

--- Circular Linked List Operations ---
1. Insert
2. Delete
3. Display
0. Back
Enter choice: 1
Enter value to insert: 20
Inserted 20 into circular list.
```



```
--- Circular Linked List Operations ---
1. Insert
2. Delete
3. Display
0. Back
Enter choice: 2
Deleted: 20

--- Circular Linked List Operations ---
1. Insert
2. Delete
3. Display
0. Back
Enter choice: 3
Circular linked list is empty.
```

Lastly, if you input 5, the program closes.



```
Choose Data Structure:
1. Stack
2. Queue
3. Linked List
4. Circular Linked List
5. Exit
Enter choice: 5
Exiting program...
```