

1. Write a program to create a singly linked list of 5 nodes where each node stores an integer. Display the contents of the list.
2. Modify the above program to allow the user to insert a new node at the beginning of the list
3. Extend the singly linked list to allow insertion of a new node at the end.
4. Write a function to delete a node from the beginning of the singly linked list and display the list after deletion.
5. Write a function to delete a node from the end of the singly linked list.
6. Write a function to insert a node at a given position (1-based index). If the position is invalid, print an error message.
7. Write a function to delete the first node that contains a specific value. Handle the case where the key is not found.
8. Implement a function that searches for a specific value in the linked list and prints its position(s)
9. Write a program to create a doubly linked list with N elements. Implement a search operation that returns the position of a given value (both forward and backward traversals)

Output

```
Doubly Linked List (forward): 10 -> 20 -> 30 -> 40 -> 50 -> NULL
Enter value to search: 20
Found at position 2 (forward)
Backward from found node: 20 <- 10 <- NULL
```

```
== Code Execution Successful ==
```

10. Implement a waiting list for a hospital using a singly linked list. Each node should store patient name and age. Implement insert at end (new patient), delete from front (patient seen), and display operations.

```
--- Hospital Waiting List Menu ---
1. Add New Patient
2. Patient Seen (Remove from Front)
3. Display Waiting List
4. Exit
Enter your choice: 1
Enter patient name: Alice
Enter age: 30

--- Hospital Waiting List Menu ---
1. Add New Patient
2. Patient Seen (Remove from Front)
3. Display Waiting List
4. Exit
Enter your choice: 3
Current Waiting List:
1. Name: Alice, Age: 30

--- Hospital Waiting List Menu ---
1. Add New Patient
2. Patient Seen (Remove from Front)
3. Display Waiting List
4. Exit
Enter your choice: 2
Patient seen: Alice, Age: 30
```