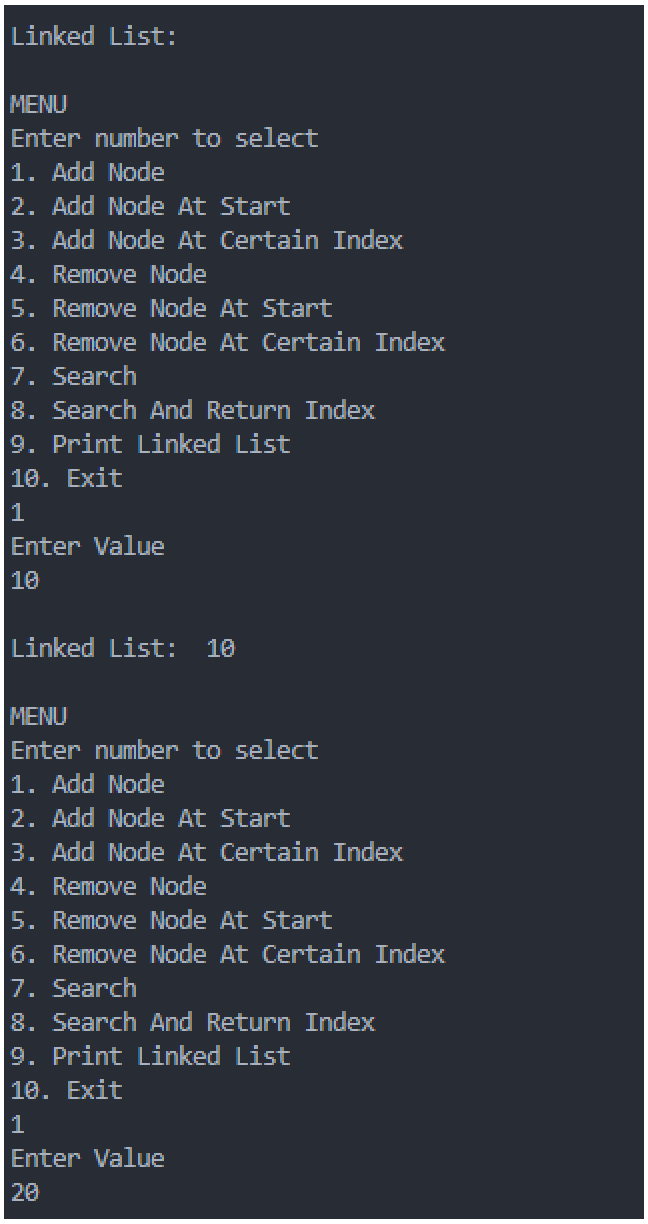
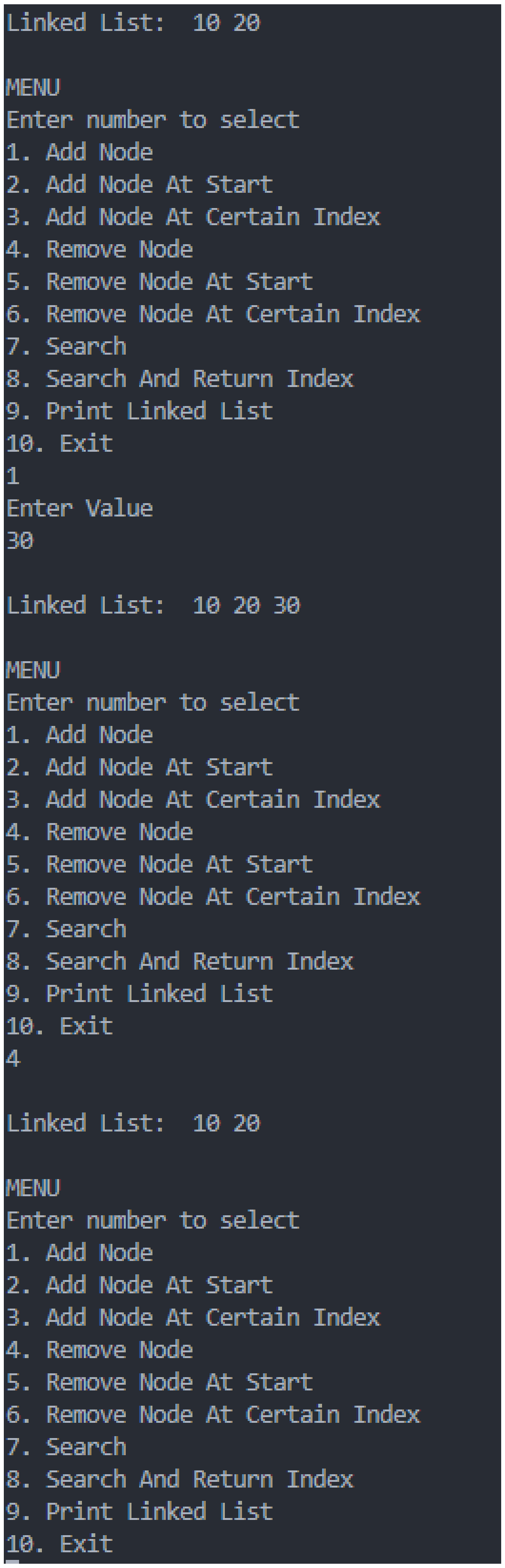
```java
import java.util.Scanner;
public class SinglyLinkedList {
    class Node {
        int val;
        Node next;
        public Node(int val){
            this.val = val;
        }
    }

    Node head = null, tail = null;

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        SinglyLinkedList list = new SinglyLinkedList();

        String menu = String.format("%nMENU%n" +
                                    "Enter number to select%n" +
                                     "1. Add Node %n" +
                                     "2. Add Node At Start%n" +
                                     "3. Add Node At Certain Index%n" +
                                     "4. Remove Node%n" +
                                     "5. Remove Node At Start%n" +
                                     "6. Remove Node At Certain Index%n" +
                                     "7. Search%n" +
                                     "8. Search And Return Index%n"+
                                     "9. Print Linked List%n"+
                                     "10. Exit" );

        int choice;
        int index;
        int target;
        do{
            list.printLinkedList();
            System.out.println(menu);
            choice = scanner.nextInt();
            int val;
            switch(choice){
                case 1:
                    System.out.println("Enter Value");
                    val = scanner.nextInt();
                    list.addNode(val);
                    break;
                case 2:
                    System.out.println("Enter Value");
                    val = scanner.nextInt();
                    list.addNode(val);

                    break;

                case 3:
                    System.out.println("Enter Value");
                    val = scanner.nextInt();
                    System.out.println("Enter index");
                    index = scanner.nextInt();
                    list.addNodeAtCertainIndex(val, index);
                    break;
                case 4:
                    list.removeNode();
                    break;
                case 5:
                    list.removeNodeAtStart();
                    break;
                case 6:
                    System.out.println("Enter index");
                    index = scanner.nextInt();
                    list.removeNodeAtCertainIndex(index);
                    break;
                case 7:
                    System.out.println("Enter value of the targer node");
                    list.search(scanner.nextInt());
                    break;
                case 8:
                    System.out.println("Enter value of the targer node");
                    list.searchAndReturnIndex(scanner.nextInt());
                    break;
                case 9:
                    list.printLinkedList();
                    break;


                }
        }while(choice != 10);

    }

    public void addNode(int val){
        if(head==null){
            Node temp = new Node(val);
            head = temp;
            tail = temp;
        }else{
            tail.next = new Node(val);
            tail = tail.next;
        }
    }

    public void addNodeAtStart(int val){
        if(head==null){
            Node temp = new Node(val);
            head = temp;
            tail = temp;
        }else{
            Node temp = new Node(val);
            temp.next = head;
            head = temp;
        }
    }




    public void addNodeAtCertainIndex(int val, int index){
            Node temp = head;
            int count = 0;
            while(temp!=null && ++count!=index)
                temp = temp.next;
            Node node  = new Node(val);
            node.next = temp.next;
        temp.next = node;
    }

//Removes the last node in the given list and updates tail node
    public void removeNode(){
         Node temp = head;
         while(temp.next!=null && temp.next.next!=null){
             temp = temp.next;
         }
         temp.next = null;
         tail = temp;
    }

// Removes the first node in the given list and updates head node
```

```java
    public void removeNodeAtStart(){
        //The first node would become zombie and should be garbage collected
after the below operation
         head = head.next;
    }

//Removes the node at the given index in the given list and updates head node
    public void removeNodeAtCertainIndex(int index){
        Node temp = head;
        int count = 0;
        while(temp!=null && ++count!=index)
            temp = temp.next;
        temp.val = temp.next.val;
        temp.next = temp.next.next;
    }

//Checks if a node with the given value exist in the list, returns true or false.
    public boolean search(int target){
        Node temp = head;
        while(temp!=null){
            if(temp.val == target)
                return true;
        }
        return false;

    }

//Checks if a node with the given value exist in the list, returns the index of
the given value in the list.
    public int searchAndReturnIndex(int target){
        Node temp = head;
        int count = 0;
        while(temp!=null){
            count++;
            if(temp.val==target) return count;
        }
        return -1;
    }




// Prints the current list
    public void printLinkedList(){
        System.out.println();
        String output = "Linked List: ";
        Node temp = head;
        while(temp!=null){
            output += " " + temp.val;
            temp = temp.next;
        }
        System.out.println(output);
    }
}
```

Output:



```
Linked List:

MENU
Enter number to select
1. Add Node
2. Add Node At Start
3. Add Node At Certain Index
4. Remove Node
5. Remove Node At Start
6. Remove Node At Certain Index
7. Search
8. Search And Return Index
9. Print Linked List
10. Exit
1
Enter Value
10

Linked List:  10

MENU
Enter number to select
1. Add Node
2. Add Node At Start
3. Add Node At Certain Index
4. Remove Node
5. Remove Node At Start
6. Remove Node At Certain Index
7. Search
8. Search And Return Index
9. Print Linked List
10. Exit
1
Enter Value
20
```



```
Linked List:  10 20

MENU
Enter number to select
1. Add Node
2. Add Node At Start
3. Add Node At Certain Index
4. Remove Node
5. Remove Node At Start
6. Remove Node At Certain Index
7. Search
8. Search And Return Index
9. Print Linked List
10. Exit
1
Enter Value
30

Linked List:  10 20 30

MENU
Enter number to select
1. Add Node
2. Add Node At Start
3. Add Node At Certain Index
4. Remove Node
5. Remove Node At Start
6. Remove Node At Certain Index
7. Search
8. Search And Return Index
9. Print Linked List
10. Exit
4

Linked List:  10 20

MENU
Enter number to select
1. Add Node
2. Add Node At Start
3. Add Node At Certain Index
4. Remove Node
5. Remove Node At Start
6. Remove Node At Certain Index
7. Search
8. Search And Return Index
9. Print Linked List
10. Exit
```