# LAB 3: Linked List 1

## Statement Purpose:

Purpose of this Lab is to familiarize the students with Linked List data structure. Another aim is to teach the students the disadvantages of using arrays and advantages of using linked lists as well as the difference between these two data structures. The students are given small tasks related to linked list which they complete during the lab session under the supervision of the lab instructor. This helps them understand the concepts well which they learn in their lectures.

## Activity Outcomes:

The students will learn how to write code for various methods which work with linked lists. They will also understand how to

- return data of second node present in the linked list
- return data of second last node present in the linked list
- delete last node in the existing linked list
- change all odd value nodes into even by subtracting 1 from every odd node and all even value nodes into odd by adding 1 into every even node.

## Theory Review (10 Minutes):

### Linked List:

A linked list is an ordered collection of data, where each element (generally called nodes) contains the location of the next element in the list. Each node essentially has two parts:

1. The **data** part. If this was a list of student records, for example, the data here may consist of a name, PID, social security number, address, phone, email, etc.
2. The **link** part. This link is used to chain the nodes together. It simply contains a reference that points to the next node in the linked list. Variable is often called “**next**”

**Disadvantages of Arrays**

- Arrays are fixed size i.e. static
- Arrays can store homogeneous (same) data
- Inserting and deleting elements is expensive because it requires shifting of remaining elements

**Advantages of Linked List**

- Linked list are not fixed size i.e. they are dynamic which means that their size can be increased or decreased during the execution of program
- Linked list can store heterogeneous (different) data
- Inserting and deleting elements is very cheap because it does not require shifting of remaining elements

**Visiting all elements of linked list**

- In case of linked list, a reference variable, generally called ***head*** points to the first node of the linked list.
- In order to visit each and every node of linked list, we need to move the reference variable (may also be called ***pointer***) from first node to the last node.
- We cannot move the ***head*** reference variable (unless it is required in order to delete the first node), otherwise we shall lose some nodes of the linked list.
- For this purpose, we take a copy of the reference variable ***head*** and call it as ***helpPtr*** which is moved from first node to the last node using ***while*** loop.
- It is depicted in the following method, which displays all nodes of a linked list.

**<u>Code for printAllNodes() Method</u>**

```
//This method is called from main() method

Public static void printAllNodes(){
     printAllNodes(head);
}

//The following method visits each and every node of linked list
and displays the value stored in data field.
```

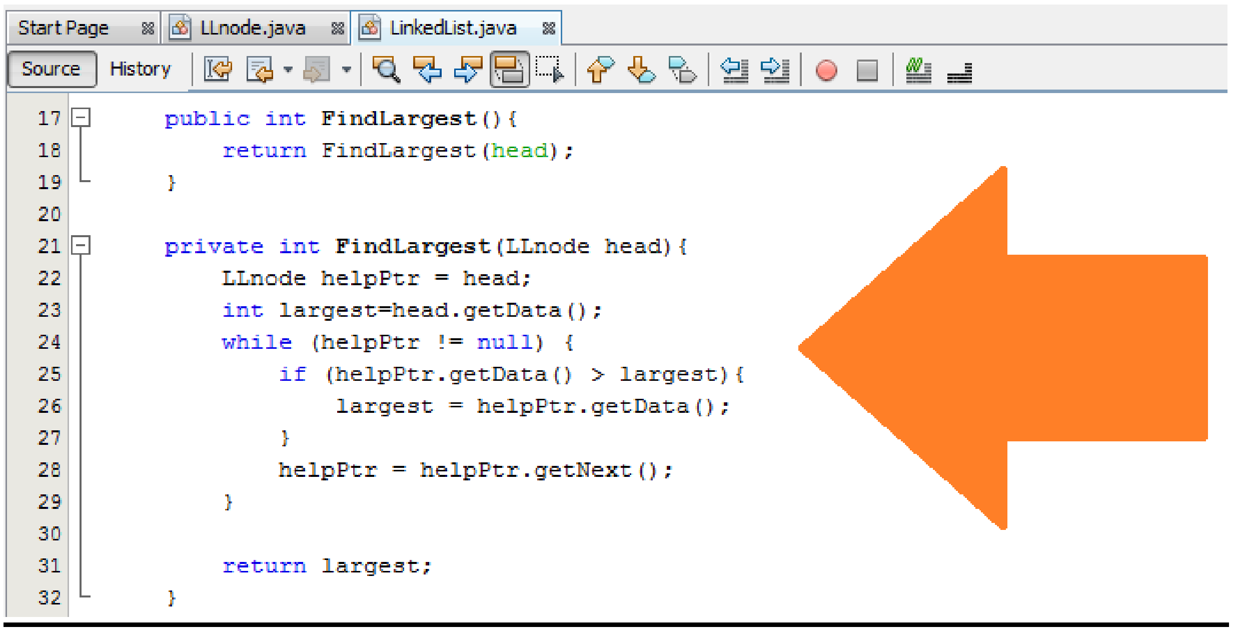
```
public static void printAllNodes(LLnode head) {
      LLnode helpPtr = head;

      while (helpPtr != null) {
            // Print the data value of the node
            System.out.print(helpPtr.getData() + ", ");
            // Moving to next node
            helpPtr = helpPtr.getNext();
      }
      System.out.println();
}
```

## Practice Activity with Lab Instructor: (10 Minutes)

Make a new project in NetBeans called LinkedListProject1, and then copy the LLnode, LinkedList, and the main LLdemo class into this project. Your job is to write code in the method FindLargest() which finds the largest node in the linked list and returns its data to the calling main() method.

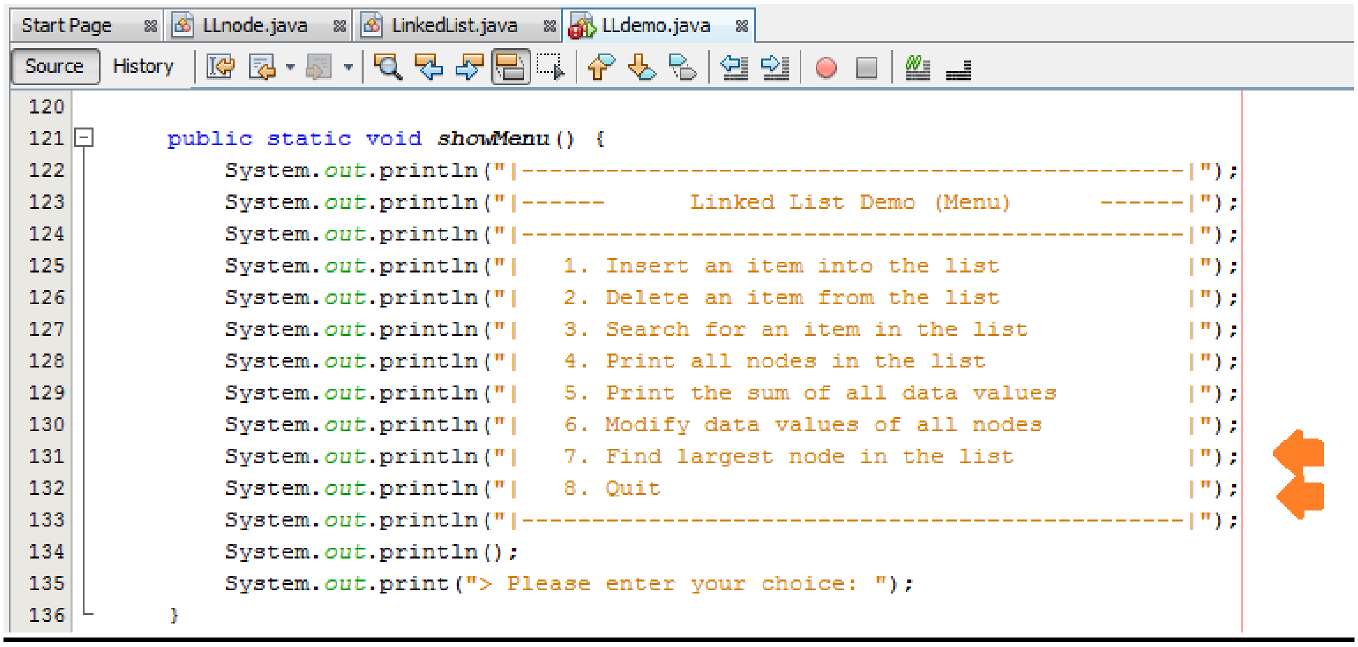
**Solution:**

1. Create a project with the name LinkListProject1
2. Create a package with the name LinkListUpdated
3. Copy LLnode class in this package.
4. Copy LinkedList class in this package and add the following method in this class.



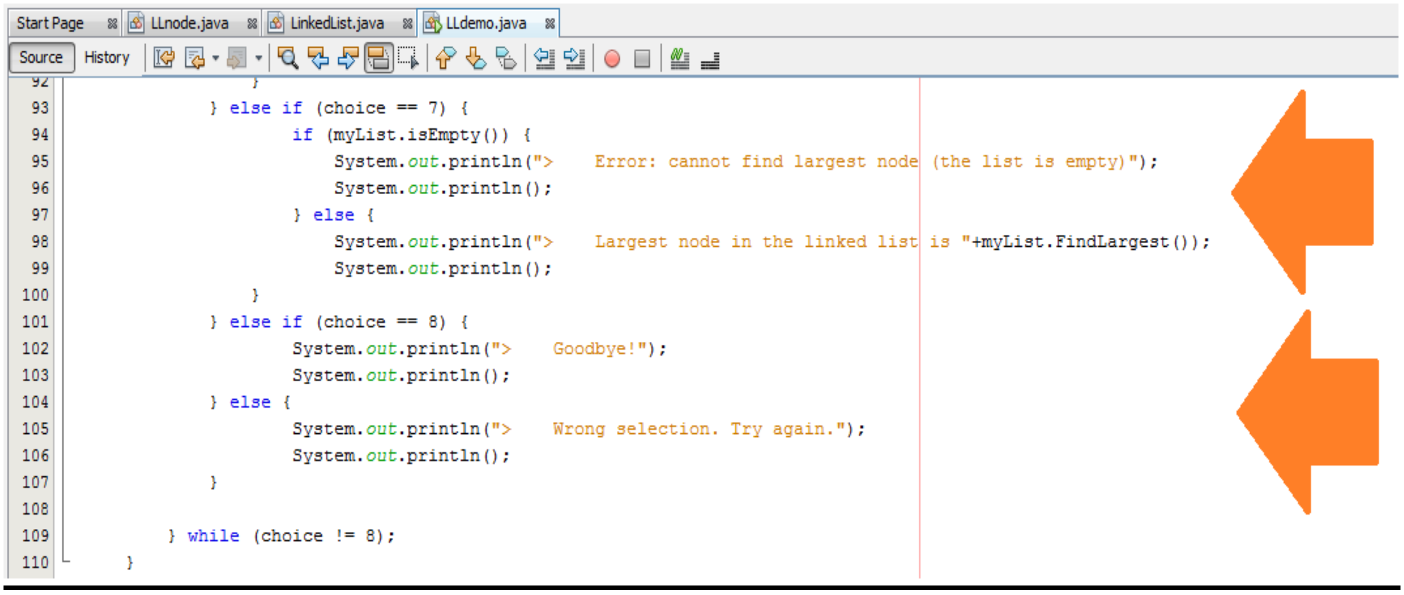
```
    public int FindLargest(){
        return FindLargest(head);
    }

    private int FindLargest(LLnode head){
        LLnode helpPtr = head;
        int largest=head.getData();
        while (helpPtr != null) {
            if (helpPtr.getData() > largest){
                largest = helpPtr.getData();
            }
            helpPtr = helpPtr.getNext();
        }

        return largest;
    }
```

5. Copy LLdemo main class in this package.
6. Add the following lines of code (shown with red arrow) in the showMenu() method of this class.



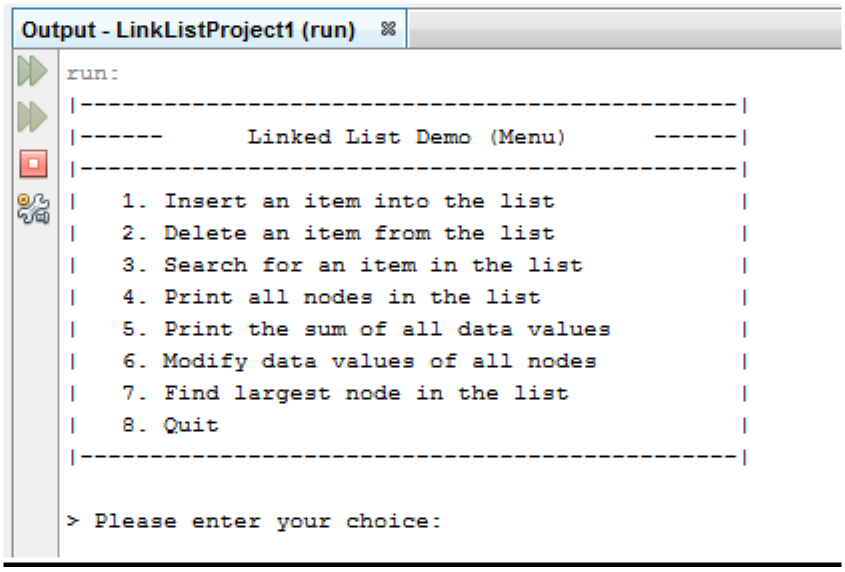
```java
    public static void showMenu() {
        System.out.println("|------------------------------------------------|");
        System.out.println("|------      Linked List Demo (Menu)       ------|");
        System.out.println("|------------------------------------------------|");
        System.out.println("|   1. Insert an item into the list              |");
        System.out.println("|   2. Delete an item from the list              |");
        System.out.println("|   3. Search for an item in the list            |");
        System.out.println("|   4. Print all nodes in the list               |");
        System.out.println("|   5. Print the sum of all data values          |");
        System.out.println("|   6. Modify data values of all nodes           |");
        System.out.println("|   7. Find largest node in the list             |");
        System.out.println("|   8. Quit                                      |");
        System.out.println("|------------------------------------------------|");
        System.out.println();
        System.out.print("> Please enter your choice: ");
    }
```

7. Add the following lines of code in the main() method of this class



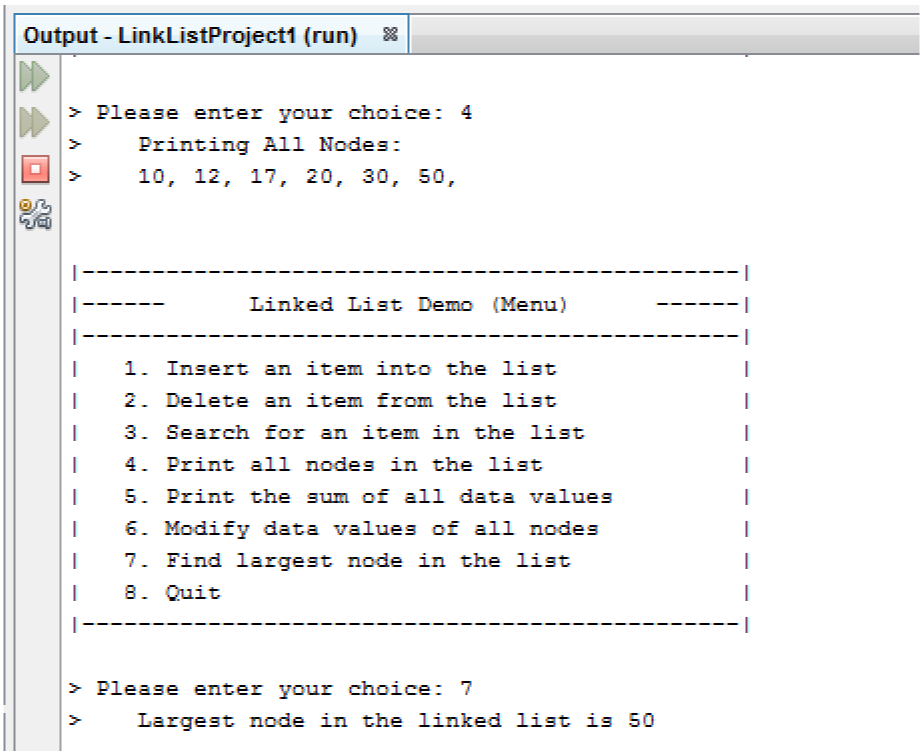
```java
                }
            } else if (choice == 7) {
                if (myList.isEmpty()) {
                    System.out.println(">    Error: cannot find largest node (the list is empty)");
                    System.out.println();
                } else {
                    System.out.println(">    Largest node in the linked list is "+myList.FindLargest());
                    System.out.println();
                }
            } else if (choice == 8) {
                System.out.println(">    Goodbye!");
                System.out.println();
            } else {
                System.out.println(">    Wrong selection. Try again.");
                System.out.println();
            }

        } while (choice != 8);
    }
```

8. **When you run this program, it will display the following menu on the screen:**


Output - LinkListProject1 (run)

```
run:
|-----------------------------------------------|
|------      Linked List Demo (Menu)      ------|
|-----------------------------------------------|
|   1. Insert an item into the list             |
|   2. Delete an item from the list             |
|   3. Search for an item in the list           |
|   4. Print all nodes in the list              |
|   5. Print the sum of all data values         |
|   6. Modify data values of all nodes          |
|   7. Find largest node in the list            |
|   8. Quit                                     |
|-----------------------------------------------|

> Please enter your choice:
```

9. **After inserting some nodes e.g. 12, 17, 10, 50, 20 and 30 in the linked list, when you select option 4 and then option 7, it will display the following output.**


Output - LinkListProject1 (run)

```
> Please enter your choice: 4
>    Printing All Nodes:
>    10, 12, 17, 20, 30, 50,


|-----------------------------------------------|
|------      Linked List Demo (Menu)      ------|
|-----------------------------------------------|
|   1. Insert an item into the list             |
|   2. Delete an item from the list             |
|   3. Search for an item in the list           |
|   4. Print all nodes in the list              |
|   5. Print the sum of all data values         |
|   6. Modify data values of all nodes          |
|   7. Find largest node in the list            |
|   8. Quit                                     |
|-----------------------------------------------|

> Please enter your choice: 7
>    Largest node in the linked list is 50
```

## LAB EXERCISES: (60 Minutes)

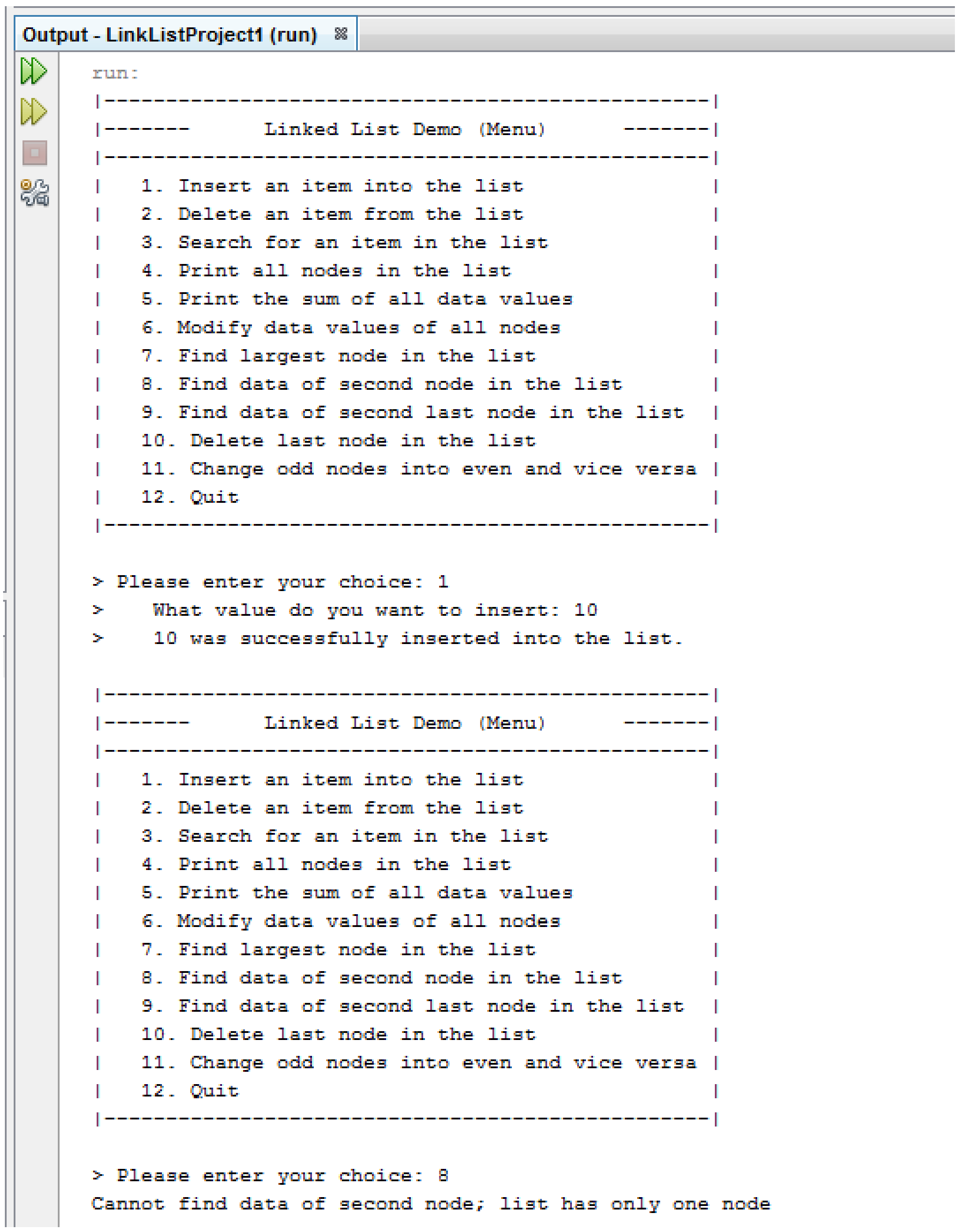
1. Add a method named Data2ndNode() in the above program which returns the data of second node in the linked list, to the calling main() method.

   **Note:** If list is empty or contains only one node, your code should display error message accordingly.

   **Hint:** Use following method header:

```
public int Data2ndNode (){
        return Data2ndNode (head);
}

private int Data2ndNode (LLnode head){

                . . . . . . . . . .

}
```

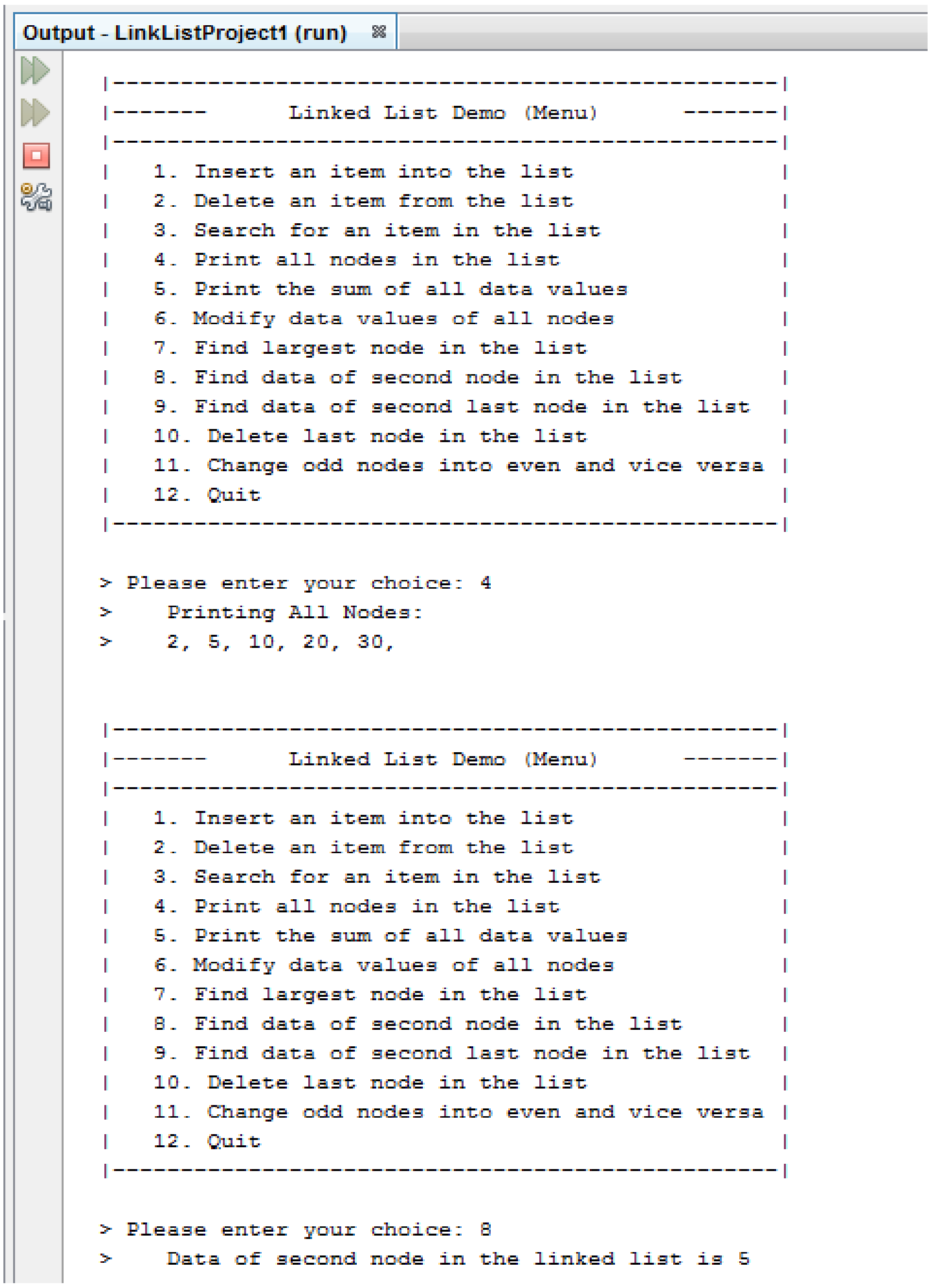
## Sample Run 1:


Output - LinkListProject1 (run)

```
run:
|-------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|-------------------------------------------------|
|   1. Insert an item into the list               |
|   2. Delete an item from the list               |
|   3. Search for an item in the list             |
|   4. Print all nodes in the list                |
|   5. Print the sum of all data values           |
|   6. Modify data values of all nodes            |
|   7. Find largest node in the list              |
|   8. Find data of second node in the list       |
|   9. Find data of second last node in the list  |
|   10. Delete last node in the list              |
|   11. Change odd nodes into even and vice versa |
|   12. Quit                                      |
|-------------------------------------------------|

> Please enter your choice: 1
>    What value do you want to insert: 10
>    10 was successfully inserted into the list.

|-------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|-------------------------------------------------|
|   1. Insert an item into the list               |
|   2. Delete an item from the list               |
|   3. Search for an item in the list             |
|   4. Print all nodes in the list                |
|   5. Print the sum of all data values           |
|   6. Modify data values of all nodes            |
|   7. Find largest node in the list              |
|   8. Find data of second node in the list       |
|   9. Find data of second last node in the list  |
|   10. Delete last node in the list              |
|   11. Change odd nodes into even and vice versa |
|   12. Quit                                      |
|-------------------------------------------------|

> Please enter your choice: 8
Cannot find data of second node; list has only one node
```

**Sample Run 2:**

If we insert nodes in the list as 10, 20, 5, 30, 2, and select option 8, it will display 5 because here the code of insertion inserts elements in ascending order. So, after arranging the elements of the list, node with data 5 becomes the second node in the list.


Output - LinkListProject1 (run)

```
|--------------------------------------------------|
|-------       Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 4
>    Printing All Nodes:
>    2, 5, 10, 20, 30,


|--------------------------------------------------|
|-------       Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 8
>    Data of second node in the linked list is 5
```

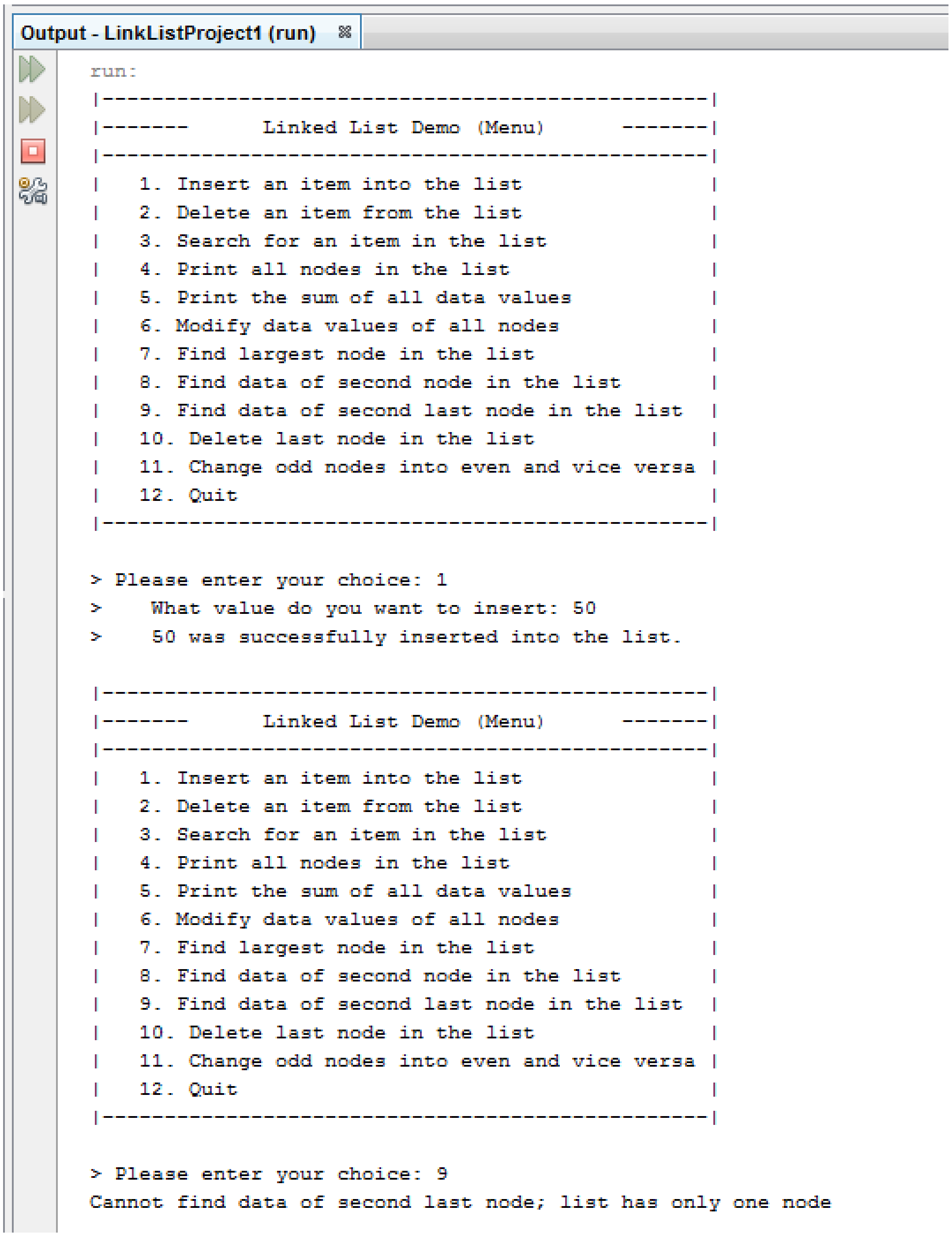
**2.** Add a method named Data2ndLastNode() in the above program which returns the data of second last node in the linked list, to the calling main() method.

**Note:** If list is empty or contains only one node, your code should display error message accordingly.

**Hint:** Use following method header:

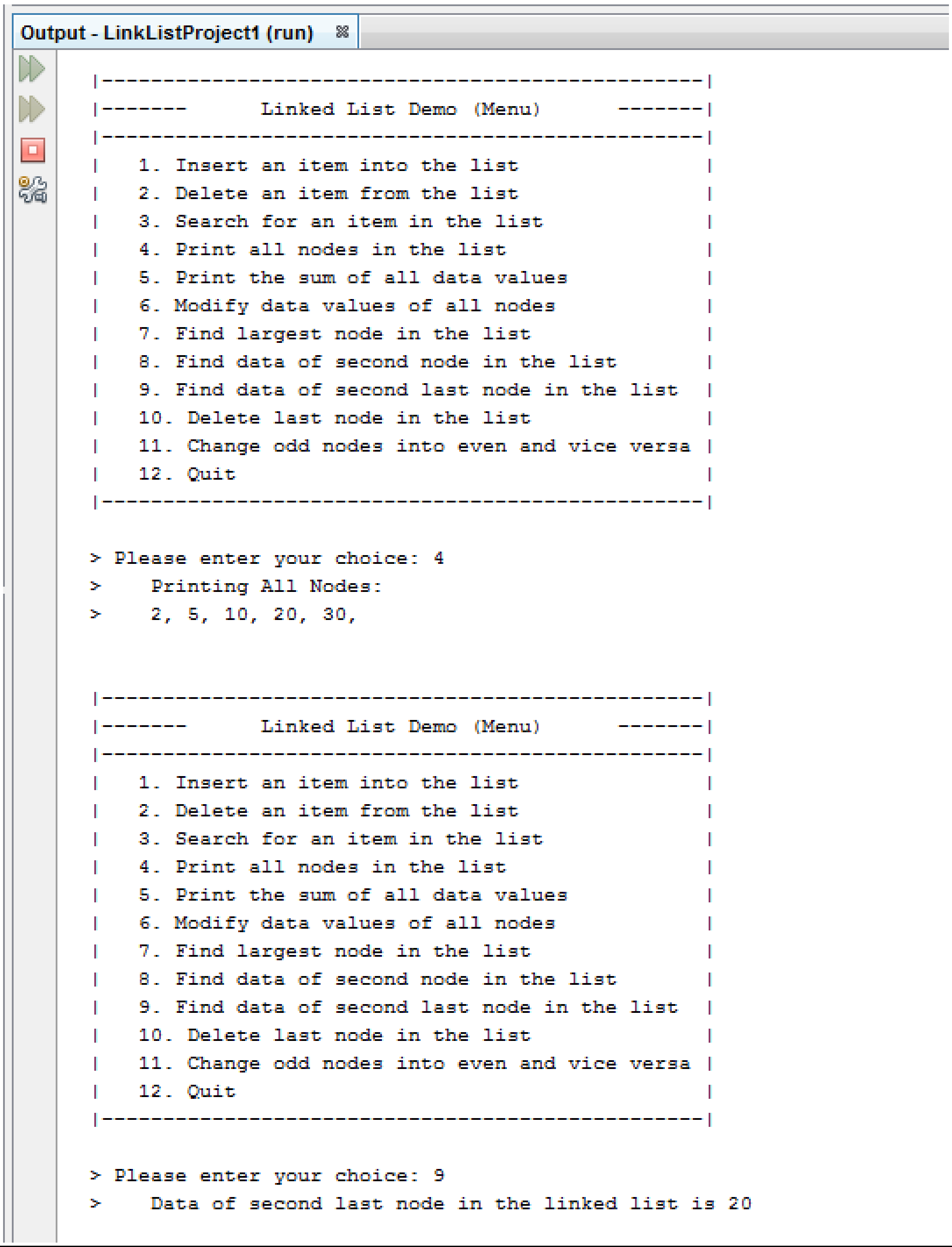
```
public int Data2ndLastNode (){
        return Data2ndLastNode (head);
}

private int Data2ndLastNode (LLnode head ){

          . . . . . . . . . .


}
```

## Sample Run 1:



```
run:
|--------------------------------------------------|
|-------       Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 1
>    What value do you want to insert: 50
>    50 was successfully inserted into the list.

|--------------------------------------------------|
|-------       Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 9
Cannot find data of second last node; list has only one node
```

**Sample Run 2:**
If we insert nodes in the list as 10, 20, 5, 30, 2, and select option 9, it will display 20 because here the code of insertion inserts elements in ascending order. So, after arranging the elements of the list, node with data 20 becomes the second last node in the list.


Output - LinkListProject1 (run)

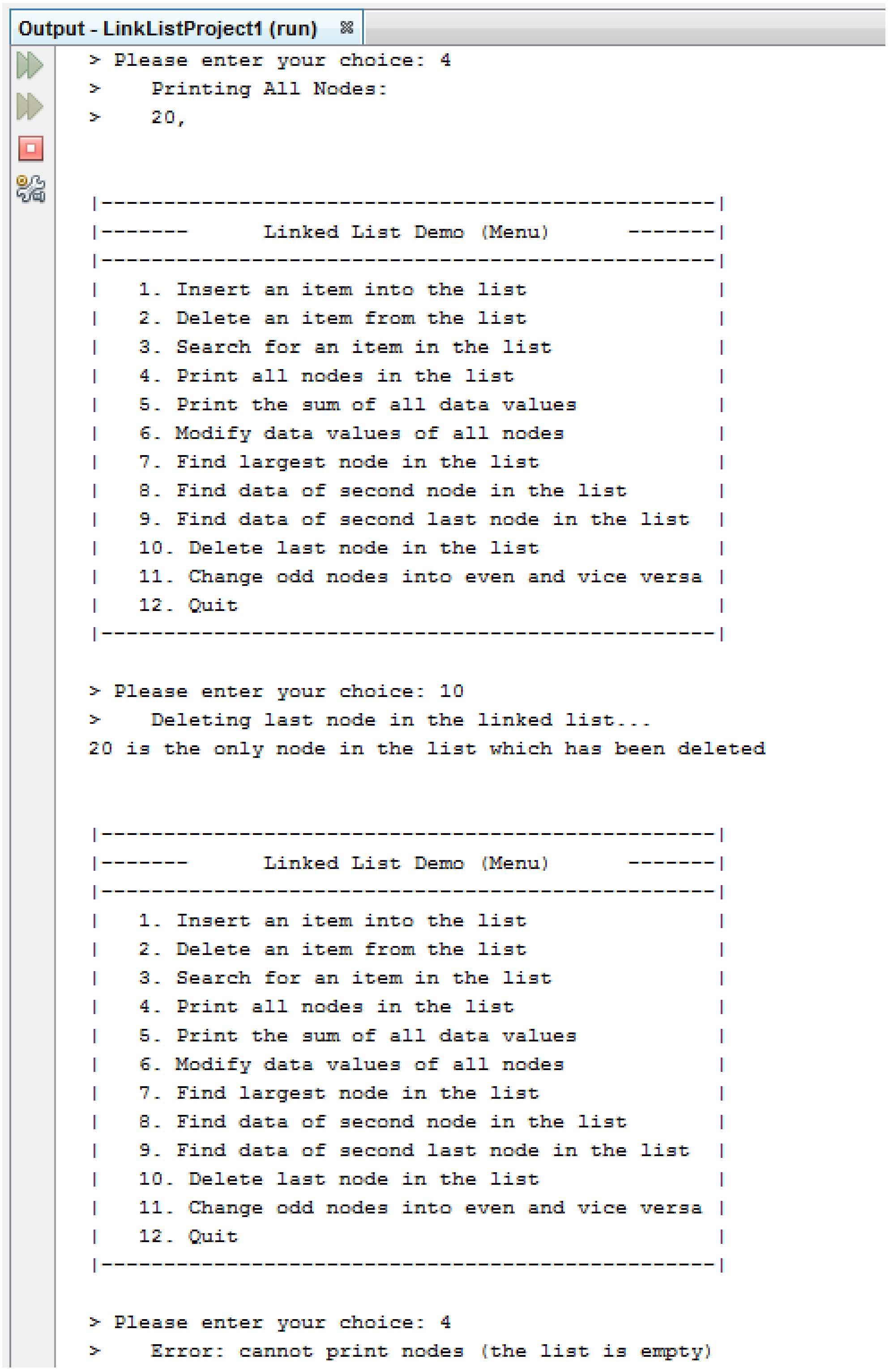
```
|-------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|-------------------------------------------------|
|   1. Insert an item into the list               |
|   2. Delete an item from the list               |
|   3. Search for an item in the list             |
|   4. Print all nodes in the list                |
|   5. Print the sum of all data values           |
|   6. Modify data values of all nodes            |
|   7. Find largest node in the list              |
|   8. Find data of second node in the list       |
|   9. Find data of second last node in the list  |
|   10. Delete last node in the list              |
|   11. Change odd nodes into even and vice versa |
|   12. Quit                                      |
|-------------------------------------------------|

> Please enter your choice: 4
>    Printing All Nodes:
>    2, 5, 10, 20, 30,


|-------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|-------------------------------------------------|
|   1. Insert an item into the list               |
|   2. Delete an item from the list               |
|   3. Search for an item in the list             |
|   4. Print all nodes in the list                |
|   5. Print the sum of all data values           |
|   6. Modify data values of all nodes            |
|   7. Find largest node in the list              |
|   8. Find data of second node in the list       |
|   9. Find data of second last node in the list  |
|   10. Delete last node in the list              |
|   11. Change odd nodes into even and vice versa |
|   12. Quit                                      |
|-------------------------------------------------|

> Please enter your choice: 9
>    Data of second last node in the linked list is 20
```

**3.** Add a method named DeleteLast() in the above program which deletes last node in the existing linked list.

**Hint:** Use following method header:

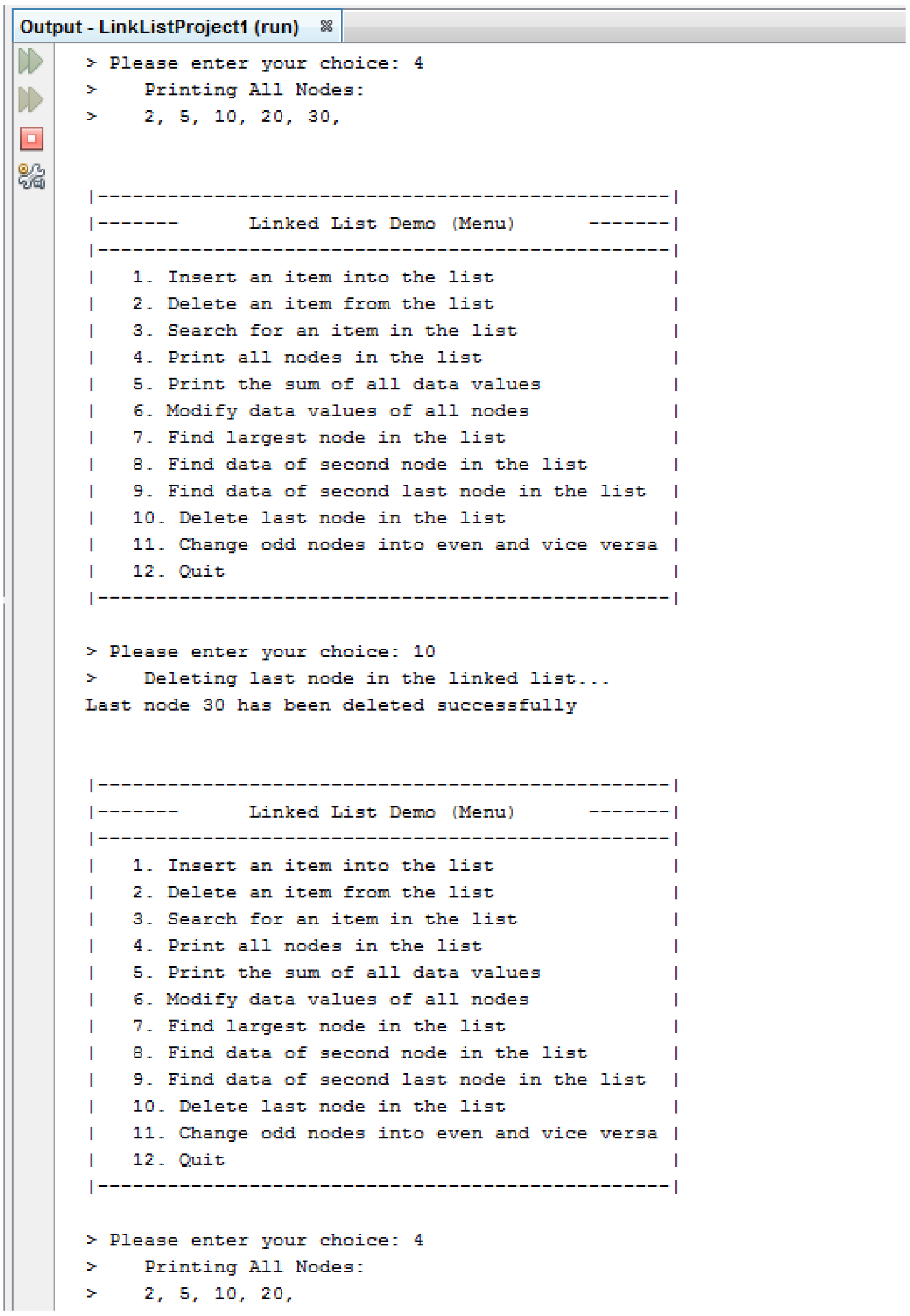
```
public void DeleteLast (){
        head = DeleteLast (head);
}

private LLnode DeleteLast (LLnode head){

                 . . . . . . . . . . .

}
```

**Sample Run 1:**



```
> Please enter your choice: 4
>    Printing All Nodes:
>    20,


|--------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 10
>    Deleting last node in the linked list...
20 is the only node in the list which has been deleted


|--------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 4
>    Error: cannot print nodes (the list is empty)
```

**Sample Run 2:**
If we insert nodes in the list as 10, 20, 5, 30, 2, and select option 10, it will delete node 30 because here the code of insertion inserts elements in ascending order. So, after arranging the elements of the list, node with data 30 becomes the last node in the list.


Output - LinkListProject1 (run)

```
> Please enter your choice: 4
>    Printing All Nodes:
>    2, 5, 10, 20, 30,


|--------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|    1. Insert an item into the list               |
|    2. Delete an item from the list               |
|    3. Search for an item in the list             |
|    4. Print all nodes in the list                |
|    5. Print the sum of all data values           |
|    6. Modify data values of all nodes            |
|    7. Find largest node in the list              |
|    8. Find data of second node in the list       |
|    9. Find data of second last node in the list  |
|    10. Delete last node in the list              |
|    11. Change odd nodes into even and vice versa |
|    12. Quit                                      |
|--------------------------------------------------|

> Please enter your choice: 10
>    Deleting last node in the linked list...
Last node 30 has been deleted successfully


|--------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|    1. Insert an item into the list               |
|    2. Delete an item from the list               |
|    3. Search for an item in the list             |
|    4. Print all nodes in the list                |
|    5. Print the sum of all data values           |
|    6. Modify data values of all nodes            |
|    7. Find largest node in the list              |
|    8. Find data of second node in the list       |
|    9. Find data of second last node in the list  |
|    10. Delete last node in the list              |
|    11. Change odd nodes into even and vice versa |
|    12. Quit                                      |
|--------------------------------------------------|

> Please enter your choice: 4
>    Printing All Nodes:
>    2, 5, 10, 20,
```

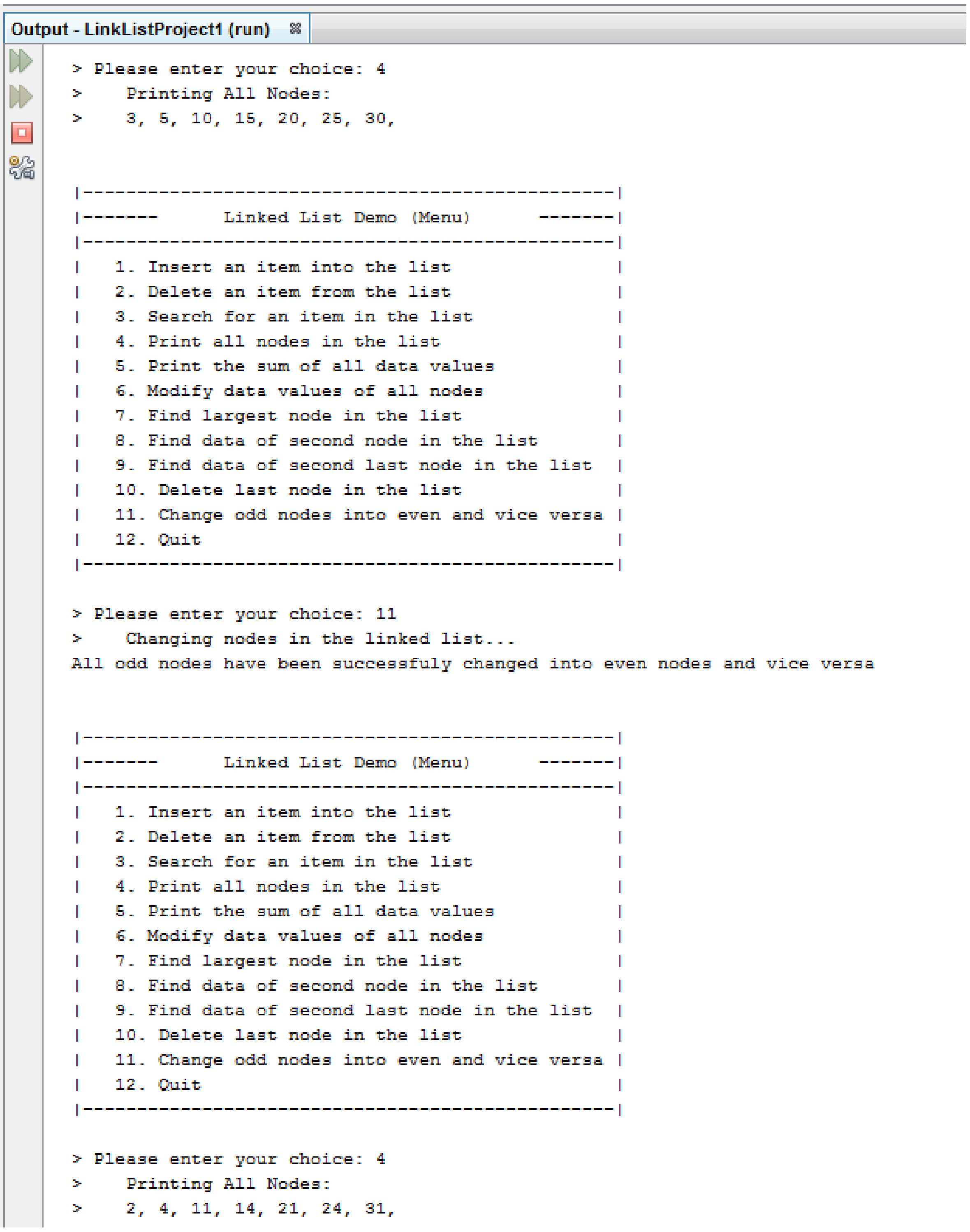
**4.** Add a method named Odd2Even2Odd() in the above program which changes all odd value nodes into even by subtracting 1 from every odd node and all even value nodes into odd by adding 1 into every even node.

**Hint:** Use following method header:

```
public void Odd2Even2Odd (){

        Odd2Even2Odd (head);

}
private void Odd2Even2Odd (LLnode head){

              . . . . . . . . . . .

}
```

**Sample Run:**

If we insert nodes in the list as 15, 20, 5, 30, 25, 10, 3 and select option 11, it will change all odd value nodes into even nodes and vice versa.



Output - LinkListProject1 (run)

```
> Please enter your choice: 4
>    Printing All Nodes:
>    3, 5, 10, 15, 20, 25, 30,


|--------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 11
>    Changing nodes in the linked list...
All odd nodes have been successfuly changed into even nodes and vice versa


|--------------------------------------------------|
|-------      Linked List Demo (Menu)      -------|
|--------------------------------------------------|
|   1. Insert an item into the list                |
|   2. Delete an item from the list                |
|   3. Search for an item in the list              |
|   4. Print all nodes in the list                 |
|   5. Print the sum of all data values            |
|   6. Modify data values of all nodes             |
|   7. Find largest node in the list               |
|   8. Find data of second node in the list        |
|   9. Find data of second last node in the list   |
|   10. Delete last node in the list               |
|   11. Change odd nodes into even and vice versa  |
|   12. Quit                                       |
|--------------------------------------------------|

> Please enter your choice: 4
>    Printing All Nodes:
>    2, 4, 11, 14, 21, 24, 31,
```