Data Structures and Algorithms

Lab Journal - Lab 2

Name: ____ABDULLAH______________________________

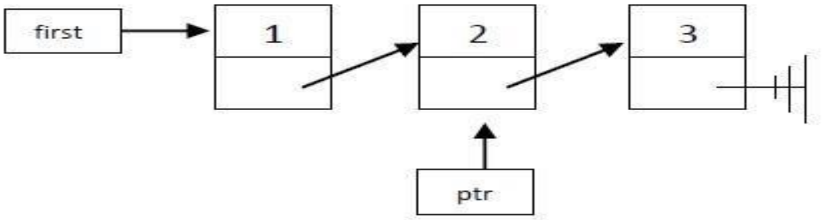
Enrollment #: ______01-134232-013____________________________

Class/Section: __________BSCS 3D________________________

Objective

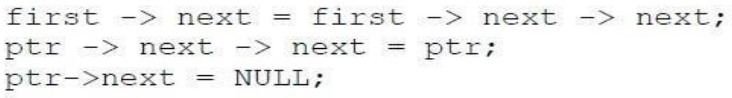
This lab session is aimed at introducing students to singly linked list. Students will be required to implement the Singly Linked List ADT. Students will also be required to create some nonmember functions to manipulate a given linked list.

Task 1: Given the following linked list, state output of the following statements.



cout<< first->data;	**1**
cout<<first->next->next->data;	**3**
cout<<ptr->next->data;	**2**

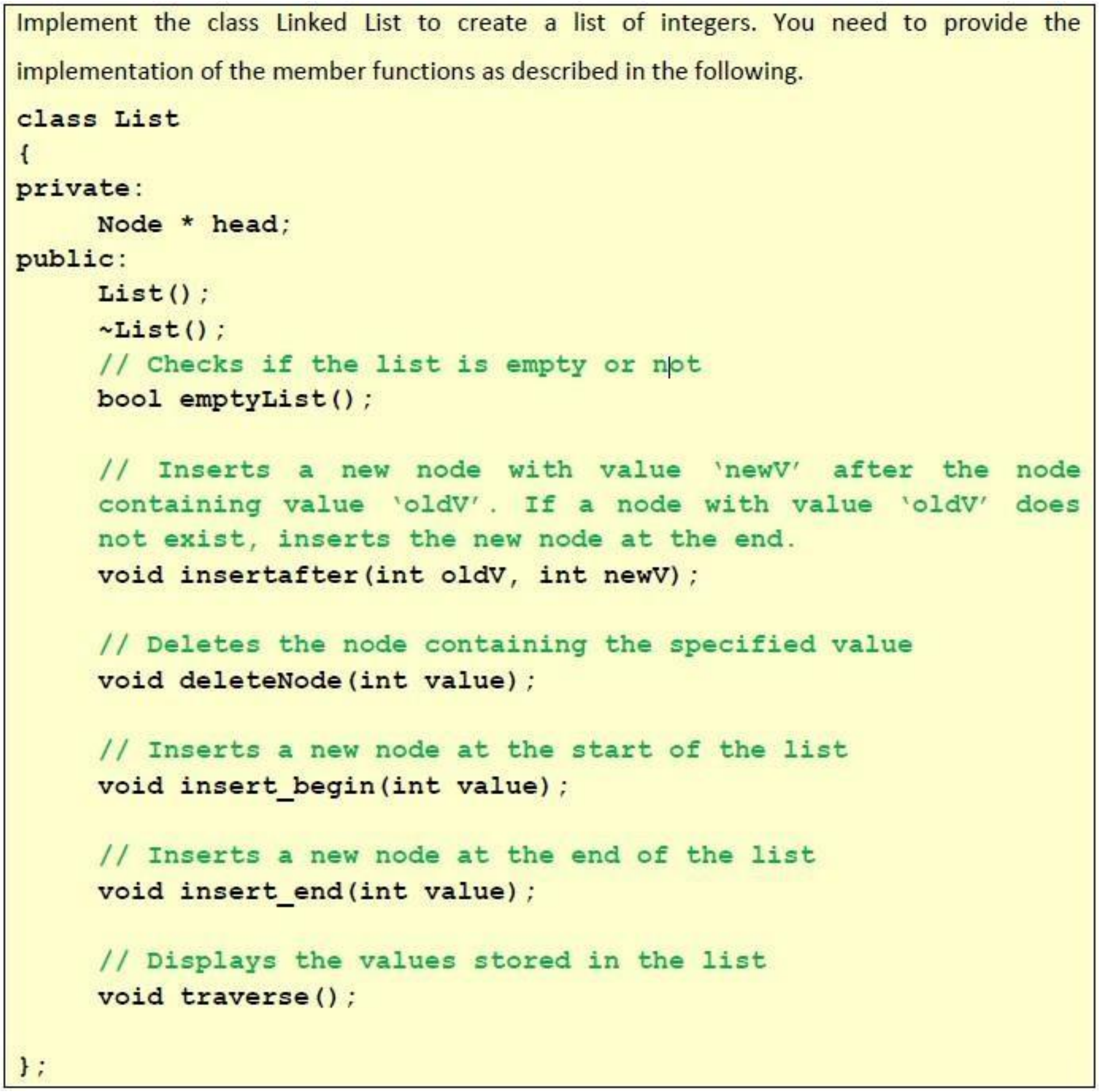
Task 2: Redraw the above list after the given instructions are executed.



```
first -> next = first -> next -> next;
ptr -> next -> next = ptr;
ptr->next = NULL;
```

```
first->1->3->2->NULL
```

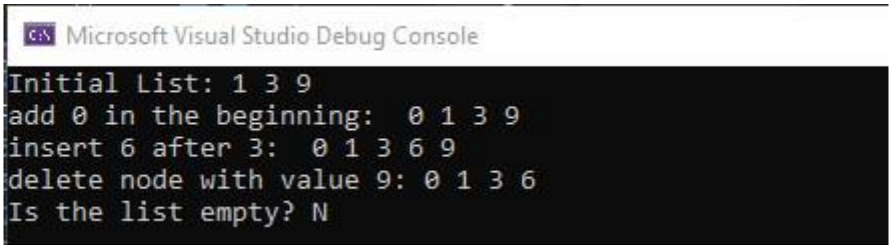
Task 3: Exercises Exercise 1 (Linked list implementation)



Implement the class Linked List to create a list of integers. You need to provide the implementation of the member functions as described in the following.

```
class List
{
private:
     Node * head;
public:
     List();
     ~List();
     // Checks if the list is empty or not
     bool emptyList();

     // Inserts a new node with value ‘newV’ after the node
     containing value ‘oldV’. If a node with value ‘oldV’ does
     not exist, inserts the new node at the end.
     void insertafter(int oldV, int newV);

     // Deletes the node containing the specified value
     void deleteNode(int value);

     // Inserts a new node at the start of the list
     void insert_begin(int value);

     // Inserts a new node at the end of the list
     void insert_end(int value);

     // Displays the values stored in the list
     void traverse();

};
```

RESULT:



```
Initial List: 1 3 9
add 0 in the beginning:  0 1 3 9
insert 6 after 3:  0 1 3 6 9
delete node with value 9: 0 1 3 6
Is the list empty? N
```

CODE:

```
#include <iostream>
using namespace std;
 class Node
{ public:
    int data;
Node* next;

    Node(int val) {         data
= val; next = NULL;
    }
};  class
LinkedList {
private:     Node*
head;
 public:
    LinkedList() { head = NULL; }

    void insert(int val) {
Node* newnode = new Node(val);
if (head == nullptr) {
head = newnode;
        }
else {
            Node* temp = head;
while (temp->next != NULL) {
temp = temp->next;
            }
            temp->next = newnode;
        }
    }
    void insertbegin(int val) {
Node* newnode = new Node(val);
newnode->next = head;         head =
newnode;
    }
    bool insertafter(int oldv, int newv) {
Node* currnode = head;
        while (currnode != NULL && currnode->data != oldv) {
currnode = currnode->next;
        }
        if (currnode == NULL) {
            cout << "Old value " << oldv << " not found in the list." << endl;
return false;
        }
        Node* newnode = new Node(newv);
newnode->next = currnode->next;
currnode->next = newnode;         return
true;
    }
    bool deletenode(int val) {
if (head == NULL) {
```

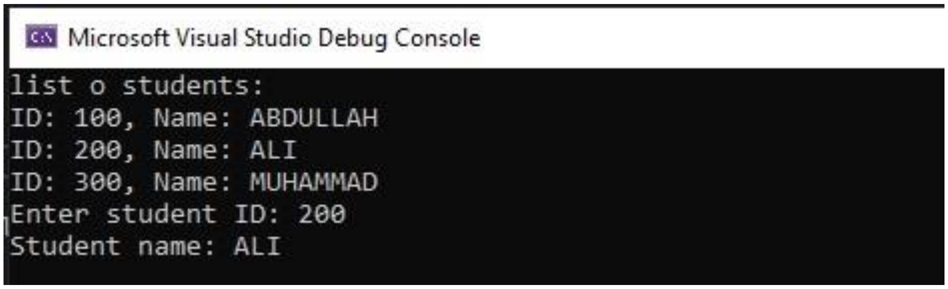
```
            cout << "List is empty." << endl;
return false;
        }
        if (head->data == val) {
Node* temp = head;
head = head->next;
delete temp;             return
true;
        }
        Node* currnode = head;
        while (currnode->next != NULL && currnode->next->data != val) {
currnode = currnode->next;
        }
        if (currnode->next == NULL) {
            cout << "Value " << val << " not found in the list." << endl;
return false;
        }

        Node* temp = currnode->next;
currnode->next = temp->next;
delete temp;         return true;
    }
    bool empty() {
return head == NULL;
    }
    void display() {         Node*
temp = head;         while (temp !=
NULL) {             cout << temp->data
<< " ";             temp = temp->next;
        }
        cout << endl;
    }

    ~LinkedList() {         while
(head != NULL) {
deletenode(head->data);
        }
    }
};
int main() {
LinkedList list;

    list.insert(1);
list.insert(3);     list.insert(9);

    cout << "Initial List: ";
list.display();

    list.insertbegin(0);
    cout << "add 0 in the beginning:  ";
list.display();
```

```
    list.insertafter(3, 6);
cout << "insert 6 after 3:  ";
list.display();

    list.deletenode(9);
    cout << "delete node with value 9: ";
list.display();

    cout << "Is the list empty? " << (list.empty() ? "Y" : "N") << endl;
return 0;
}
```

Exercise 2 (Storing Records in a List)

Modify the above list ADT to create a list of students, where each node of the list contains student ID and name. Alongwith, basic list functions, also write a non-member function that allows the user to search a student in a list by using his ID and then displays the name of the student, if found.

RESULT:



```
list o students:
ID: 100, Name: ABDULLAH
ID: 200, Name: ALI
ID: 300, Name: MUHAMMAD
Enter student ID: 200
Student name: ALI
```

CODE:

```
#include <iostream>
#include <string> using
namespace std;
 class Student {
private:     int
id;     string
name;     Student*
next;
 public:

    Student(int id, string name) {
this->id = id;          this->name
= name;          next = NULL;
    }      int
getId() {
return id;
```

```
    }      string
getName() {
return name;
    }

    Student* getNext() {
return next;
    }
    void setNext(Student* nextstudent) {
next = nextstudent;
    }
};  class
StudentList {
private:
Student* head;
 public:

    StudentList() { head = NULL; }

    void insert(int id, string name) {
Student* newstudent = new Student(id, name);
if (head == NULL) {             head = newstudent;
        }
else {
            Student* temp = head;
while (temp->getNext() != NULL) {
temp = temp->getNext();
            }
            temp->setNext(newstudent);
        }
}
    void searchById(int id) {
Student* temp = head;         while
(temp != NULL) {              if (temp-
>getId() == id) {
                cout << "Student name: " << temp->getName() << endl;
return;             }
            temp = temp->getNext();
        }
        cout << "error: student not found" << endl;
    }
    void displaystudents() {
Student* temp = head;         while
(temp != NULL) {
            cout << "ID: " << temp->getId() << ", Name: " << temp->getName() << endl;
            temp = temp->getNext();
        }
    }
};

int main() {
StudentList List;
```

```
    List.insert(100, "ABDULLAH");
    List.insert(200, "ALI");
    List.insert(300, "MUHAMMAD");

    cout << "list o students: " << endl;


    List.displaystudents();

    int searchId;
    cout << "Enter student ID: ";
cin >> searchId;
List.searchById(searchId);

    return 0;
}
```

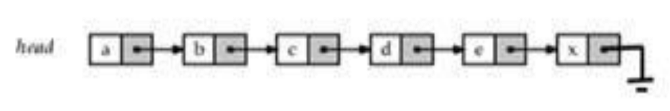
Write a C++ program which allows users to create a list of integers and the n provides them with choice to call following user- defined function s:

a. A function which accepts a pointer to the head of the list and returns the maximum value in the list.

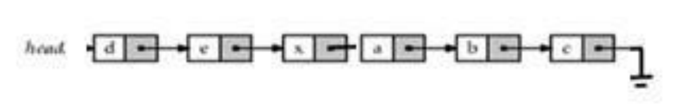
b. A function that counts and returns the total number of nodes in the list .

c. A function that prints only the even - numbered (not even - valued) nodes of the list.

d. A function that splits the given list in half and swaps the two halves in the following manner.



Initial

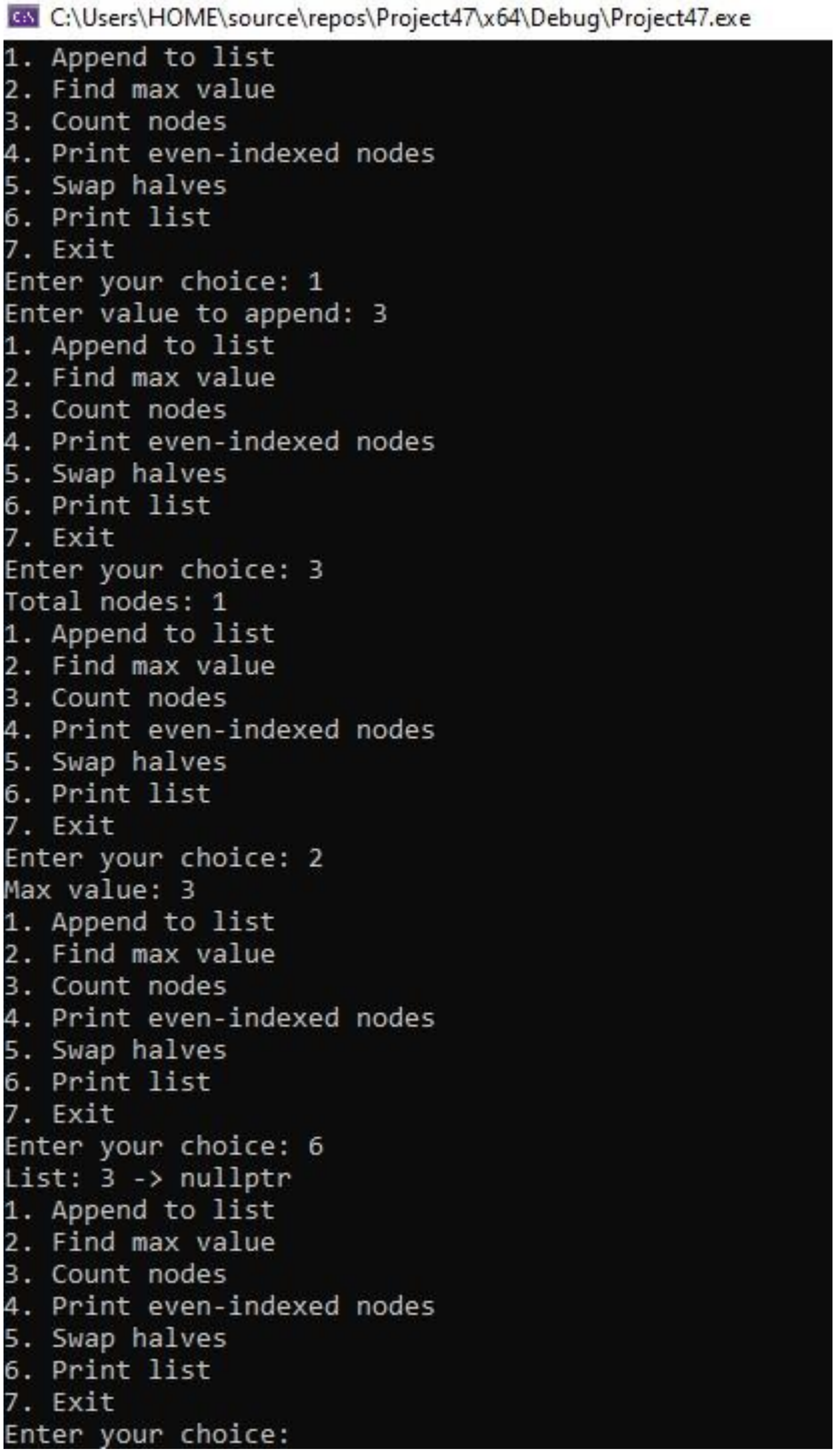


After

*Note: Create these as non- member functions.

Exercise 3 (List Traversal based Utility Functions)

RESULT:



C:\Users\HOME\source\repos\Project47\x64\Debug\Project47.exe

```
1. Append to list
2. Find max value
3. Count nodes
4. Print even-indexed nodes
5. Swap halves
6. Print list
7. Exit
Enter your choice: 1
Enter value to append: 3
1. Append to list
2. Find max value
3. Count nodes
4. Print even-indexed nodes
5. Swap halves
6. Print list
7. Exit
Enter your choice: 3
Total nodes: 1
1. Append to list
2. Find max value
3. Count nodes
4. Print even-indexed nodes
5. Swap halves
6. Print list
7. Exit
Enter your choice: 2
Max value: 3
1. Append to list
2. Find max value
3. Count nodes
4. Print even-indexed nodes
5. Swap halves
6. Print list
7. Exit
Enter your choice: 6
List: 3 -> nullptr
1. Append to list
2. Find max value
3. Count nodes
4. Print even-indexed nodes
5. Swap halves
6. Print list
7. Exit
Enter your choice:
```

CODE:

```
#include <iostream>
using namespace std;
```

```
 struct Node {
int data;
Node* next;
```

```
    Node(int val) : data(val), next(nullptr) {}
};
void append(Node*& head, int value) {
Node* newNode = new Node(value);
if (!head) {         head = newNode;
return;
    }

    Node* temp = head;
while (temp->next) {
temp = temp->next;
    }
    temp->next = newNode;
}
int findMax(Node* head) {     if
(!head) return INT_MIN;      int
maxVal = head->data;     Node*
temp = head->next;     while
(temp) {         if (temp->data >
maxVal) {             maxVal =
temp->data;
        }         temp =
temp->next;
    }     return
maxVal;
}
int countNodes(Node* head) {
int count = 0;     Node*
temp = head;     while
(temp) {         count++;
temp = temp->next;
    }     return
count;
}
void printEvenNodes(Node* head) {
Node* temp = head;     int index = 0;
while (temp) {         if (index % 2
== 0) {              cout << temp-
>data << " ";
        }         temp =
temp->next;
index++;     }     cout <<
endl;
}
void swapHalves(Node*& head) {     if
(!head || !head->next) return;
Node* slow = head, * fast = head, *
prev = nullptr;

    while (fast && fast->next) {
prev = slow;         slow =
slow->next;         fast = fast-
>next->next;
```

```
    }

    prev->next = nullptr;
Node* firstHalf = head;
    Node* secondHalf = slow;


    Node* prevNode = nullptr;
    Node* currNode = secondHalf;
Node* nextNode;     while
(currNode) {         nextNode =
currNode->next;         currNode-
>next = prevNode;         prevNode
= currNode;         currNode =
nextNode;
    }
    secondHalf = prevNode;

    head = secondHalf;
Node* temp = head;     while
(temp->next) {         temp
= temp->next;
    }
    temp->next = firstHalf;
}
void printList(Node* head) {
while (head) {
        cout << head->data << " -> ";
head = head->next;
    }
    cout << "nullptr" << endl;
}
int main() {
    Node* head = nullptr;
int choice, value;
     do
{
        cout << "1. Append to list\n2. Find max value\n3. Count nodes\n4. Print
even-indexed nodes\n5. Swap halves\n6. Print list\n7. Exit\n";         cout <<
"Enter your choice: ";         cin >> choice;

        switch (choice) {
case 1:
            cout << "Enter value to append: ";
cin >> value;
            append(head, value);             break;         case
2:             cout << "Max value: " << findMax(head) << endl;
break;         case 3:             cout << "Total nodes: " <<
countNodes(head) << endl;             break;         case 4:
cout << "Even-indexed nodes: ";
printEvenNodes(head);             break;         case 5:
swapHalves(head);             cout << "Halves swapped." << endl;
break;         case 6:
```

```
            cout << "List: ";             printList(head);
break;         case 7:             cout << "Exiting program." <<
endl;             break;         default:             cout <<
"Invalid choice. Please try again." << endl;
        }
    } while (choice != 7);

    while (head) {
Node* temp = head;
head = head->next;
delete temp;
    }
    return 0;
}
```

Implement the given exercises and get them checked by your instructor.

S No.	**Exercise**	**Checked By:**
1.	Exercise 1	
2.	Exercise 2	
3.	Exercise 3a	
4.	Exercise 3b	
5.	Exercise 3c	
6.	Exercise 3d	

+++++++++++++++++++++++++++