Name: Muhammad Zohaib khan

Reg# BSE203003

Subject: DS LAB 6

Section:1

Date:2 -12-2021

```
#include<iostream>
using namespace std;
class Node {
public:
	int data;
	Node *next;
	Node()
	{
		data = 0;
		next = NULL;

	}
	Node(int x)
	{
		data = x;
		next = NULL;
	}
	int getData()
	{
		return data;
	}
	void setData(int x)
	{
		data = x;
	}
	Node* getNext()
	{
		return next;

	}
	void setnext(Node* n)
	{
		next = n;
	}
};
class linked_list {
	Node *head;
public:
	linked_list()
	{
		head = NULL;
	}
	int addNodeatFront(Node *n)
	{
		int i = 0;
		n->next = head;
		head = n;
		i = 1;
		return i;
	}
	int isEmpty()
	{
		if (head == NULL)
		{
			return 1;
		}
		else
		{
			return 0;

		}
	}
```

```
Node* getLastNode()
{
	Node *ptr = head;
	while (ptr->next != NULL)
	{
		ptr = ptr->next;
	}
	return ptr;
}
Node* search(int k)
{
	Node *ptr = head;
	int i = 0;
	if (head == NULL)
	{
		cout << "List is empty";
		return NULL;

	}
	else
	{
		while (ptr->next != NULL && ptr->getData() != k)
		{
			ptr = ptr->getNext();
			if (ptr->getData() == k)
			{
				i = 1;
				break;

			}
		}
		if (i == 0)
		{
			return NULL;
		}
		else {
			return ptr;
		}
	}
}
Node* deleteFirstNode()
{
	if (head == NULL)
	{
		cout << "list is empty";
		return NULL;
	}
	else {
		Node *temp = head;
		head = head->next;
	}
}
Node* deleteLastNode()
{
	if (head == NULL)
	{
		cout << "list is empty";
		return NULL;
	}
	else
	{
		Node *temp = head;
```

```
                while (temp->next == NULL)
                {
                        delete temp;
                }
        }
}
void insertAfter(Node *prev_node, int new_data)
{
        if (prev_node == NULL)
        {
                cout << "The given previous can Not be NULL";

        }
        else {
                Node *new_node = new Node();
                new_node->data = new_data;
                new_node->next - prev_node->next;
                prev_node->next = new_node;
        }
}
Node* deleteNode(Node* n)
{
        Node *ptr = head;
        if (head == NULL)
        {
                cout << "list is empty";
                return NULL;
                delete ptr;
                return head;
        }
        else
        {
                if (ptr == n)
                {
                        ptr->setnext(n->getNext());
                        return n;

                }
                else
                {
                        while (ptr->getNext() != n)
                        {
                                ptr = ptr->getNext();
                        }
                        ptr->setnext(n->getNext());
                }return n;
        }
}
int addNodeatEnd(Node* n)
{
        if (head == NULL)
        {
                cout << "list is empty";
                return NULL;
        }
        else
        {
                Node *n1 = getLastNode();
                n1->next = n;
                cout << "After inserting last value is";
                getLastNode();
        }
```

```
                return 1;

        }
        void display()
        {
                Node *ptr = head;
                if (head == NULL)
                {
                        cout << "list is empty";
                }
                else
                {
                        while (ptr != NULL)
                        {
                                cout << "Data:" << ptr->data << endl;
                                ptr = ptr->next;
                        }
                }
        }

};
int main()
{
        int x, q, ch = 0, num, s;
        linked_list l;
        int add, del, option;
        Node *n3 = new Node();
        Node *n1 = new Node();
        Node *a1 = new Node();
        do {
                while (ch != 7)
                {
                        cout << "Press 1 Insert Front" << endl;
                        cout << "Press 2 Insert End" << endl;
                        cout << "Press 3 Insert after number" << endl;
                        cout << "Press 4 Delete front" << endl;
                        cout << "Press 5 Delete End" << endl;
                        cout << "Press 6 Delete any Number from List" << endl;
                        cout << "Press 7 Display all element" << endl;
                        cin >> ch;
                        switch (ch) {
                        case 1:
                                cout << "Enter number of elements you want to enter";
                                cin >> num;
                                for (int i = 0; i < num; i++)
                                {
                                        cout << i + 1 << "Enter value" << endl;
                                        cin >> x;
                                        Node *n4 = new Node(x);
                                        q = l.addNodeatFront(n4);

                                }
                                if (q == 1)
                                {
                                        cout << "Inserted" << endl;
                                        l.display();
                                }
                                break;
                        case 2:
                                n3 = l.getLastNode();
                                cout << "Last node=" << n3->getData() << endl;
                                break;
```

```
                        case 3:
                                cout << "Enter value you wnat to enter";
                                cin >> num;
                                cout << "Enter number after which you want to enter";
                                cin >> add;

                                a1->data = add;
                                l.insertAfter(a1, num);
                                break;
                        case 4:
                                l.deleteFirstNode();
                                break;
                        case 5:
                                l.deleteLastNode();
                                break;
                        case 6:
                                cout << "Enter number you want to delate";
                                cin >> s;
                                n3 = l.search(s);
                                if (n3 == NULL)
                                {
                                        cout << "match not found" << endl;
                                }
                                else
                                {
                                        cout << "match found" << endl;
                                        l.deleteNode(n3);

                                }
                                break;
                        case 7:
                                l.display();
                                break;
                        }cout << "Enter 1 to run program again";
                        cin >> option;
                }
        } while (option == 1);
}
```

```
C:\Users\zohai\source\repos\Project1\Debug\Project1.exe
Press 1 Insert Front
Press 2 Insert End
Press 3 Insert after number
Press 4 Delete front
Press 5 Delete End
Press 6 Delete any Number from List
Press 7 Display all element
1
Enter number of elements you want to enter5
1Enter value
1
2Enter value
2
3Enter value
3
4Enter value
4
5Enter value
5
Inserted
Data:5
Data:4
Data:3
Data:2
Data:1
Enter 1 to run program again1
Press 1 Insert Front
Press 2 Insert End
Press 3 Insert after number
Press 4 Delete front
Press 5 Delete End
Press 6 Delete any Number from List
Press 7 Display all element
4
Enter 1 to run program again1
Press 1 Insert Front
Press 2 Insert End
Press 3 Insert after number
Press 4 Delete front
Press 5 Delete End
Press 6 Delete any Number from List
Press 7 Display all element
7
Data:4
Data:3
Data:2
Data:1
Enter 1 to run program again1
```

Practice Task 2:

```
#include<iostream>
using namespace std;
class Node {
public:
```

```
        int data;
        Node *next;
        Node()
        {
                data = 0;
                next = NULL;

        }
        Node(int x)
        {
                data = x;
                next = NULL;
        }
        int getData()
        {
                return data;
        }
        void setData(int x)
        {
                data = x;
        }
        Node* getNext()
        {
                return next;

        }
        void setnext(Node* n)
        {
                next = n;
        }
};
class linked_list {
        Node *head;
public:
        linked_list()
        {
                head = NULL;
        }
        int addNodeatFront(Node *n)
        {
                int i = 0;
                n->next = head;
                head = n;
                i = 1;
                return i;
        }
        int isEmpty()
        {
                if (head == NULL)
                {
                        return 1;
                }
                else
                {
                        return 0;

                }
        }
        Node* getLastNode()
        {
                Node *ptr = head;
                while (ptr->next != NULL)
```

```
	{
		ptr = ptr->next;
	}
	return ptr;
}
Node* search(int k)
{
	Node *ptr = head;
	int i = 0;
	if (head == NULL)
	{
		cout << "List is empty";
		return NULL;

	}
	else
	{
		while (ptr->next != NULL && ptr->getData() != k)
		{
			ptr = ptr->getNext();
			if (ptr->getData() == k)
			{
				i = 1;
				break;

			}
		}
		if (i == 0)
		{
			return NULL;
		}
		else {
			return ptr;
		}
	}
}
Node* deleteFirstNode()
{
	if (head == NULL)
	{
		cout << "list is empty";
		return NULL;
	}
	else {
		Node *temp = head;
		head = head->next;
	}
}
Node* deleteLastNode()
{
	if (head == NULL)
	{
		cout << "list is empty";
		return NULL;
	}
	else
	{
		Node *temp = head;
		while (temp->next == NULL)
		{
			delete temp;
		}
```

```
        }
}
void insertAfter(Node *prev_node, int new_data)
{
        if (prev_node == NULL)
        {
                cout << "The given previous can Not be NULL";

        }
        else {
                Node *new_node = new Node();
                new_node->data = new_data;
                new_node->next - prev_node->next;
                prev_node->next = new_node;
        }
}
Node* deleteNode(Node* n)
{
        Node *ptr = head;
        if (head == NULL)
        {
                cout << "list is empty";
                return NULL;
                delete ptr;
                return head;
        }
        else
        {
                if (ptr == n)
                {
                        ptr->setnext(n->getNext());
                        return n;

                }
                else
                {
                        while (ptr->getNext() != n)
                        {
                                ptr = ptr->getNext();
                        }
                        ptr->setnext(n->getNext());
                }return n;
        }
}
int addNodeatEnd(Node* n)
{
        if (head == NULL)
        {
                cout << "list is empty";
                return NULL;
        }
        else
        {
                Node *n1 = getLastNode();
                n1->next = n;
                cout << "After inserting last value is";
                getLastNode();
        }
        return 1;

}
void display()
```

```
	{
		Node *ptr = head;
		if (head == NULL)
		{
			cout << "list is empty";
		}
		else
		{
			while (ptr != NULL)
			{
				cout << "Data:" << ptr->data << endl;
				ptr = ptr->next;
			}
		}
	}
	void Remove_Duplicate()
	{
		Node *ptr = head;
		Node *temp;
		Node *dup;
		while (ptr != NULL && ptr->next != NULL)
		{
			temp = ptr;
			while (temp->next != NULL)
			{
				if (ptr->data == temp->next->data)
				{
					dup = temp->next;
					temp->next = temp->next->next;
					delete dup;

				}
				else {
					temp = temp->next;
				}
			}ptr = ptr->next;
		}

	}
	void count()
	{
		int count = 0;
		Node *current = head;
		while (current != NULL)
		{
			count++;
			current = current->next;
		}cout << "Total numbers of Nodes currently present in this list are :"
<< count << endl;
	}
};
int main()
{
	int x, q, ch = 0, num, s;
	linked_list l;
	int add, del, option;
	Node *n3 = new Node();
	Node *n1 = new Node();
	Node *a1 = new Node();
	do {
		while (ch != 7)
		{
```

```
cout << "Press 1 Insert Front" << endl;
cout << "Press 2 Insert End" << endl;
cout << "Press 3 Insert after number" << endl;
cout << "Press 4 Delete front" << endl;
cout << "Press 5 Delete End" << endl;
cout << "Press 6 Delete any Number from List" << endl;
cout << "Press 7 Display all element" << endl;
cout << "Press 8 to Remove Duplicate" << endl;
cout << "Press 9 to Count Nodes" << endl;
cin >> ch;
switch (ch) {
case 1:
	cout << "Enter number of elements you want to enter";
	cin >> num;
	for (int i = 0; i < num; i++)
	{
		cout << i + 1 << "Enter value" << endl;
		cin >> x;
		Node *n4 = new Node(x);
		q = l.addNodeatFront(n4);

	}
	if (q == 1)
	{
		cout << "Inserted" << endl;
		l.display();
	}
	break;
case 2:
	n3 = l.getLastNode();
	cout << "Last node=" << n3->getData() << endl;
	break;
case 3:
	cout << "Enter value you wnat to enter";
	cin >> num;
	cout << "Enter number after which you want to enter";
	cin >> add;

	a1->data = add;
	l.insertAfter(a1, num);
	break;
case 4:
	l.deleteFirstNode();
	break;
case 5:
	l.deleteLastNode();
	break;
case 6:
	cout << "Enter number you want to delate";
	cin >> s;
	n3 = l.search(s);
	if (n3 == NULL)
	{
		cout << "match not found" << endl;
	}
	else
	{
		cout << "match found" << endl;
		l.deleteNode(n3);

	}
	break;
```

```
                        case 7:
                               l.display();
                               break;
                        case 8:
                               l.Remove_Duplicate();
                               break;
                        case 9:
                               l.count();
                        }cout << "Enter 1 to run program again";
                        cin >> option;
                }
        } while (option == 1);
}
```

C:\Users\zohai\source\repos\Project1\Debug\Project1.exe

```
3
5Enter value
3
6Enter value
4
Inserted
Data:4
Data:3
Data:3
Data:2
Data:1
Data:1
Enter 1 to run program again1
Press 1 Insert Front
Press 2 Insert End
Press 3 Insert after number
Press 4 Delete front
Press 5 Delete End
Press 6 Delete any Number from List
Press 7 Display all element
Press 8 to Remove Duplicate
Press 9 to Count Nodes
9
Total numbers of Nodes currently present in this list are :6
Enter 1 to run program again1
Press 1 Insert Front
Press 2 Insert End
Press 3 Insert after number
Press 4 Delete front
Press 5 Delete End
Press 6 Delete any Number from List
Press 7 Display all element
Press 8 to Remove Duplicate
Press 9 to Count Nodes
8
Enter 1 to run program again1
Press 1 Insert Front
Press 2 Insert End
Press 3 Insert after number
Press 4 Delete front
Press 5 Delete End
Press 6 Delete any Number from List
Press 7 Display all element
Press 8 to Remove Duplicate
Press 9 to Count Nodes
7
Data:4
Data:3
Data:2
Data:1
Enter 1 to run program again
```

Practice Task 3:

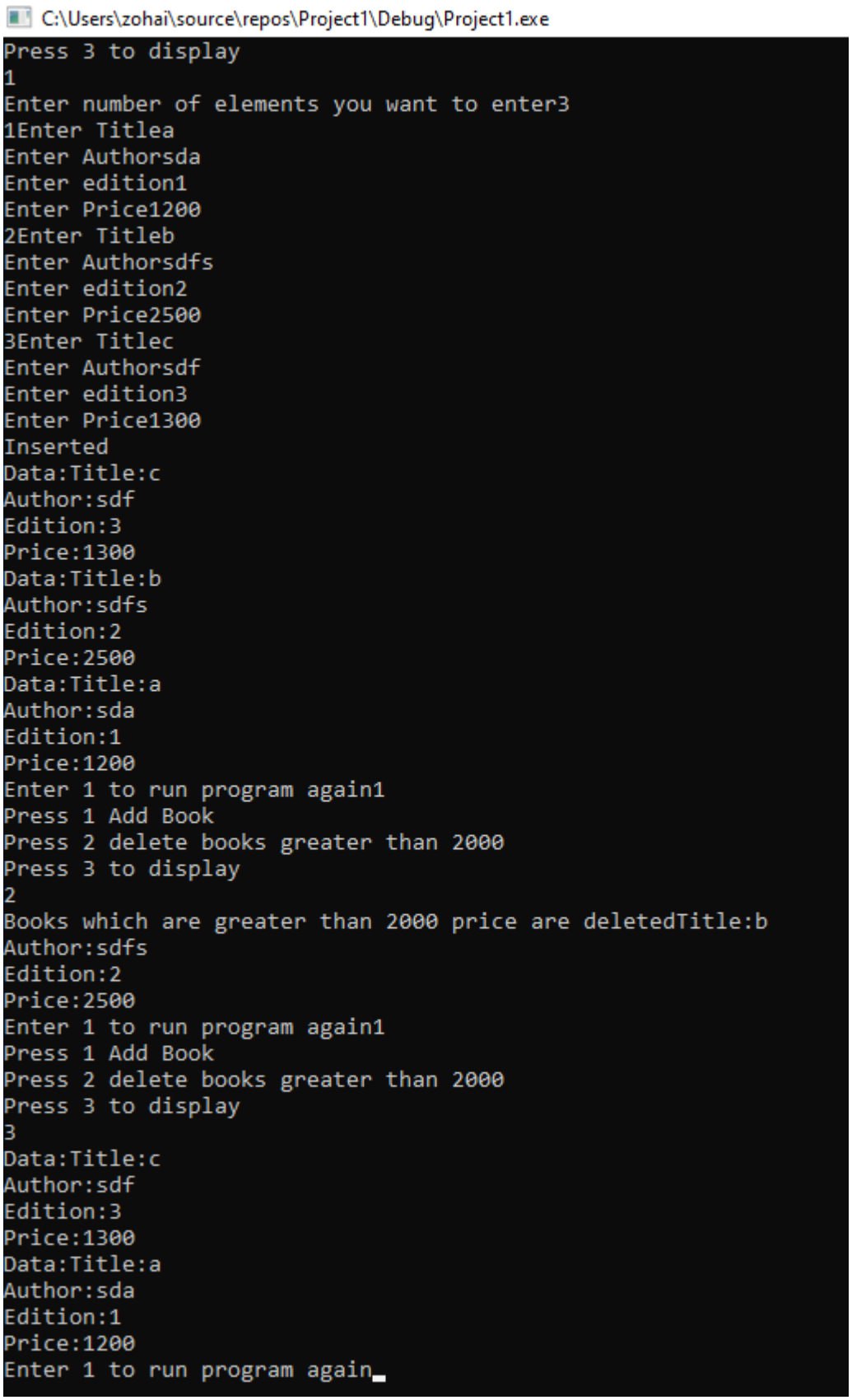
```
#include<iostream>
#include<string>
using namespace std;
```

```
class Node {
public:
	string title;
	string author;
	int edition;
	int price;
	Node *next;
	Node()
	{

		next = NULL;

	}
	Node(string t, string a, int e, int pr)
	{
		title = t;
		author = a;
		edition = e;
		price = pr;
		next = NULL;
	}
	void getData()
	{
		cout << "Title:" << title << endl;
		cout << "Author:" << author << endl;
		cout << "Edition:" << edition << endl;
		cout << "Price:" << price << endl;
	}
	void setData(string t, string a, int e, int pr)
	{
		title = t;
		author = a;
		edition = e;
		price = pr;
	}

	void setnext(Node* n)
	{
		next = n;
	}
	Node* getNext()
	{
		return next;

	}
};
class linked_list {
	Node *head;
public:
	linked_list()
	{
		head = NULL;
	}
	int addbook(Node *n)
	{
		int i = 0;
		n->next = head;
		head = n;
		i = 1;
		return i;
	}
```

```
Node* deleteNode(Node* n)
{
	Node *ptr = head;
	if (head == NULL)
	{
		cout << "list is empty";
		return NULL;
		delete ptr;
		return head;
	}
	else
	{
		if (ptr == n)
		{
			ptr->setnext(n->getNext());
			return n;

		}
		else
		{
			while (ptr->getNext() != n)
			{
				ptr = ptr->getNext();
			}
			ptr->setnext(n->getNext());
		}return n;
	}
}




void display()
{
	Node *ptr = head;
	if (head == NULL)
	{
		cout << "list is empty";
	}
	else
	{
		while (ptr != NULL)
		{
			cout << "Data:";
			ptr->getData();
			ptr = ptr->next;
		}
	}
}
Node* search()
{
	Node *p = head;
	int i = 0;
	if (head == NULL)
	{
		cout << "List is empty";
		return NULL;

	}
	else
```

```
		{
			while (p->next != NULL)
			{

				if (p->price > 2000)
				{
					i = 1;
					break;

				} p = p->getNext();
			}
				if (i == 0)
				{
					return NULL;
				}
				else if(i==1){
					return p;
					i = 0;
				}
			}
		}


};
int main()
{
	int  q, ch = 0, num;
	linked_list l;
	string t, a;
	int pri, e;
	int   option;
	Node *n4;

	do {
		while (ch != 4)
		{
			cout << "Press 1 Add Book" << endl;
			cout << "Press 2 delete books greater than 2000" << endl;
			cout << "Press 3 to display" << endl;

			cin >> ch;
			switch (ch) {
			case 1:
				cout << "Enter number of elements you want to enter";
				cin >> num;
				for (int i = 0; i < num; i++)
				{
					cout << i + 1;
					cout << "Enter Title";
					cin >> t;
					cout << "Enter Author";
					cin >> a;
					cout << "Enter edition";
					cin >> e;
					cout << "Enter Price";
					cin >> pri;
					Node *n4 = new Node(t,a,e,pri);
					q = l.addbook(n4);


				}
				if (q == 1)
```

```
                        {
                                cout << "Inserted" << endl;
                                l.display();
                        }
                        break;



                case 2:
                        cout << "Books which are greater than 2000 price are
deleted";
                        n4 = l.search();
                        if(n4==NULL)
                        {
                                cout << "No books deleted";
                        }
                        else {
                                n4->getData();
                                l.deleteNode(n4);
                        }
                        break;
                case 3:
                        l.display();
                        break;
                }cout << "Enter 1 to run program again";
                cin >> option;
        }
    } while (option == 1);
}
```


C:\Users\zohai\source\repos\Project1\Debug\Project1.exe
Press 3 to display
1
Enter number of elements you want to enter3
1Enter Titlea
Enter Authorsda
Enter edition1
Enter Price1200
2Enter Titleb
Enter Authorsdfs
Enter edition2
Enter Price2500
3Enter Titlec
Enter Authorsdf
Enter edition3
Enter Price1300
Inserted
Data:Title:c
Author:sdf
Edition:3
Price:1300
Data:Title:b
Author:sdfs
Edition:2
Price:2500
Data:Title:a
Author:sda
Edition:1
Price:1200
Enter 1 to run program again1
Press 1 Add Book
Press 2 delete books greater than 2000
Press 3 to display
2
Books which are greater than 2000 price are deletedTitle:b
Author:sdfs
Edition:2
Price:2500
Enter 1 to run program again1
Press 1 Add Book
Press 2 delete books greater than 2000
Press 3 to display
3
Data:Title:c
Author:sdf
Edition:3
Price:1300
Data:Title:a
Author:sda
Edition:1
Price:1200
Enter 1 to run program again