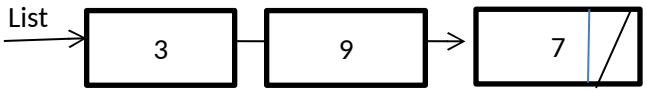
Data Structures, Handout 8, Linked List Insertion and Deletion

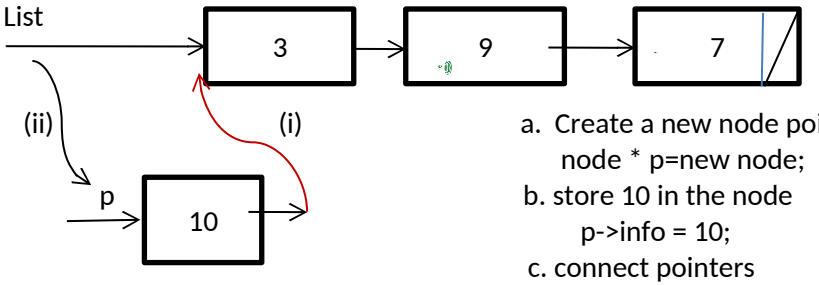
a. Insertion

Given the following linked list and declarations for each node



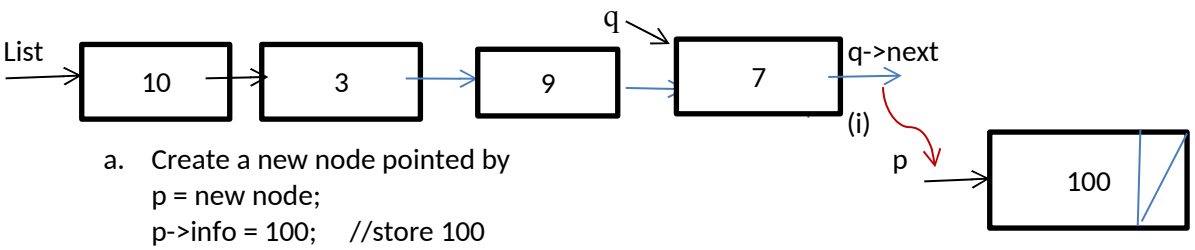

```
struct node {
        int    info;
        node  *next;
};
 node *List;
```

Insert 10 in front of the list



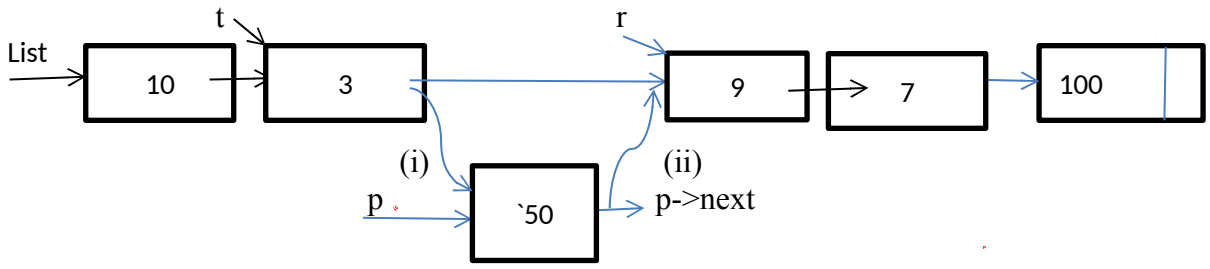
a. Create a new node pointed by p
   node * p=new node;
b. store 10 in the node
   p->info = 10;
c. connect pointers
   (i) p->next =List;
   (ii) List = p;

Insert 100 at the rear of the list



a. Create a new node pointed by
   p = new node;
   p->info = 100; //store 100
   p->next= NULL; //set the next to NULL
b. Connect the last node to this new node. We need to have a pointer to point to the last nod
   node *q=List;
   while(q->next != NULL)
   { q=q->next; }
   q->next = p; // connect the last node to the node pointed by p (i)

Insert 50 before the node whose info is 9



a. Create a node pointed by p, set the info to 50
   p=new node; p->info = 50;
b. Find two pointers t and r so that r points to the node with info 9 make t to follow r
   node *t = List; node *r = List;
   while(r->info != 9)
   { t = r; r = r->next; }
c. Connect pointers
   (i) t->next = p;
   (ii) p->next = r;

1. Display the content of all nodes

```
node *p= List; // make a copy of the master pointer
while( p != NULL )
{
    cout<<p->info<<"-- >"; p=p->next;  //output: 10→3→50→9→7→100→NULL
}
cout<<"NULL\n";
```

2. Count the number of nodes

```
int counter=0;
p = List; //use pointer p to visit all nodes
while( p != NULL)
{ counter++; p=p->next; }
cout<<counter;  //output: 6
```

3. Find the maximum info

```
p=List;
int max=p->info;//suppose the first node holds the max info
p=p->next; //start from the next node
while( p!= NULL)
{
    if (p->info > max ) max= p->info; p=p->next;
}
cout<<"Maximum info="<<max<<endl;
```

4. Display the last node

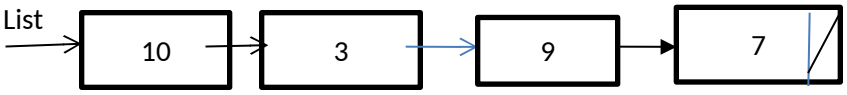
```
//to access the info of the last node we need a pointer to point to the last node
p = List;
//make p to point to the last node
while( p->next != NULL)
    p=p->next;
cout<<p->info;  //output: 100
```

5. Display the max and the min info

```
p=List;
int max=min = p->info; // suppose the first info is max and min
p=p->next; // start from the second node
while( p!= NULL) // visit all nodes
{
    if (p->info > max ) max=p->info;
    if (p->info < min ) min = p->info;
   p = p-> next;
}
cout<<Maximum info ="<<max<<"  and minimum info ="<<min;
```

c. **Deletion** or removing node from a linked list

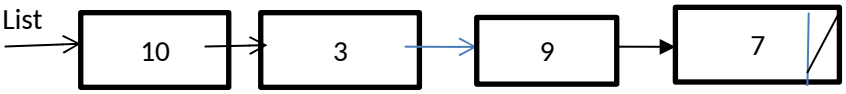
Given the following linked list and declaration



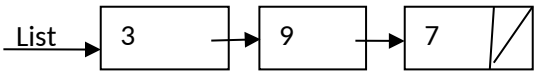

```
struct node
{  int info;
   node *next;
};
node *List;
```

Delete the first node

To delete any node you need a pointer to point to that node

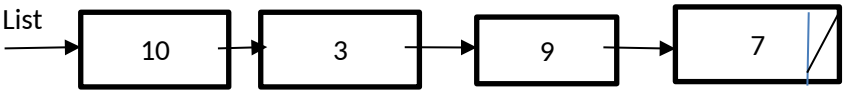


a. Create a pointer p to point to the first node
   node *p = List;
b. Make List to point to the next node
   List = p->next;



c. Delete the first node pointed by p.
   We use the predefined function delete to do it
   delete p; //the node does not exist anymore, nut you can reuse p

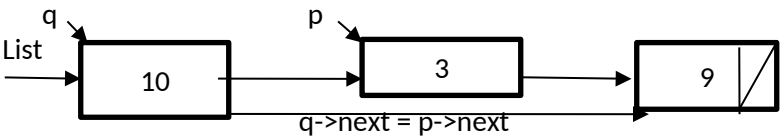
Delete the last node



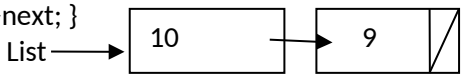
To delete the last node we need a pointer p to point to it. After deleting last node the node with info will become the last node and we should set its next to NULL, so we make q to point to the node with info 9.

a. node *p=List; node *q=List;// create p and q
b. // make p to point to the last node and q point to node with info 9 (q will follow p)
   while(p->next != NULL)
   { q = p; p = p->next; }
c. delete p; //delete the last node
   q->next=NULL; //make the node with info 9 the new last node

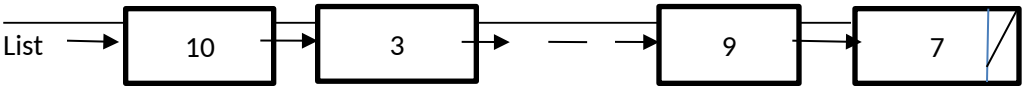
Delete the node with info 3



a. Make p point to the node with info 3, and have q to follow p
   p=q=List;
   while (p->info != 3) { q=p; p=p->next; }
b. Connect pointers
   q->next = p->next; delete p;



This function deletes the first node of linked list pointed by List




```
// calling statement: deleteNode( List );

//function:  void deleteNode( node& *p)// initially p=List, after deleting the first node
                                        //List shoud point to the next node too
        {
           node *q=p;
            p=p->next; //skip the first node, p now is pointing to the next node and
                       //we have to pass this change to calling statement. The & in the
                       //function heading does the job
           delete p;
        }
```

For linked list which are stack or queue, this function act as pop operation

The linked list implementation of stack, queue, and ordered linked list.

Note; Given data : 5 9 and 3

a. Insert each new item in front of the linked list
   i. Insert 5: list→5
   ii. Insert 9: list→9 →5
   iii. Insert 3: list →3 → 9 → 5
   Display the list : 3 9 5 which is the reverse of the original data, this linked list behaves like **stack** (Last In First Out(LIGO) of First In Last Out (FILO))
b. Insert each new item at the rear of the list
   i. Insert 5: list →5
   ii. Insert 9: list →5 →9
   iii. Insert 3: list 5 → 9 → 3
   Display the list 5 9 3 which is the same order as the original data, this linked list behaves like a ***queue***(First In First Out (FIFO))
c. Insert each new item in the list and keep the list ordered (from lowest to highest)
   i. Insert 5: list →5
   ii. Insert 9: list →5 →9
   iii. Insert 3: list 3→ 5 →9
   Display the list 3 5 9 which is the ordered form of the original data, this linked list called and **ordered** linked list.

Pointer implementations of stack, queue, and ordered linked list

a. Creating pointer implementation of stack

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

template <class T>
class STACK
{
private:
	struct node
	{
		T    info;
		node *next;
	};
	node *stack;
public:
	STACK() { stack = NULL; }// constructore
	bool notEmptyStack()
	{
		return (stack == NULL) ? true : false;
	}
	void pushStack( T x)
	{
		//insert x in front of the list
		node *p = new node; p->info = x;
		p->next = stack; stack = p;
	}
	void displayStack()
	{
		node *p = stack;
		while (p != NULL)
		{
			cout << p->info << "-->"; p = p->next;
		}
		cout << "NULL\n";
	}
	T popStack()
	{
		//return theinfo of the first node and then
		//delete that node
		T popedElement;
		node *p = stack;
		popedElement = p->info;
		stack = p->next;
		delete p;

		return popedElement;
	}
};

int main()
{
	int a[5] = { 5,9,2,7,3 };
	//display a in reverse order
	STACK<int>   s;
	for (int i = 0; i < 5; ++i)
```

```
            s.pushStack(a[i]);
      // display the stack
      // if you need to use this stack in the program
      // again, then call displayStack, otherwise call the popStack
      cout << "This is array a in reverse oreder \n";
      s.displayStack();
      // or use the popStack if you are not going to use
      //the stack you just created
      cout << " This is the reverse of array a using pop\n";
      while (!s.notEmptyStack())
      {
            int x = s.popStack();
            cout << x << "-->";
      }
      cout << "NULL\n";
      system("pause");
      return 0;
}
/*-----------------------output-----------------
This is array a in reverse oreder
3-->7-->2-->9-->5-->NULL
This is the reverse of array a using pop
3-->7-->2-->9-->5-->NULL
Press any key to continue . . .
-------------------------------------------------*/
```

b. Creating Pointer Implementation of queue

```
//creating pointer implementation of quue
#include <iostream>
using namespace std;

template <class T>
class QUEUE
{
private:
      struct node
      {
            T    info;
            node *next;
      };
      node *queue;
public:
      QUEUE() { queue = NULL; }// constructore
      bool emptyQueue()
      {
            return (queue == NULL) ? true : false;
      }
      void pushQueue( T x)//
      {
            //insert x at the rear of the list
            node* r = new node;
            r->info = x; r->next = NULL;
            if (queue == NULL)
                  queue = r;
            else
            {
```

```
                        //make p to point at the rear node
                        node* p = queue;
                        while (p->next != NULL)
                                p = p->next;
                        p->next = r;
                }
        }
        void displayQueue()
        {
                node *p = queue;
                while (p != NULL)
                {
                        cout << p->info << "-->"; p = p->next;
                }
                cout << "NULL\n";
        }
        T popQueue()
        {
                //return the info of the first node and then
                //delete that node
                T popedElement;
                node *p = queue;
                popedElement = p->info;
                queue = p->next;
                delete p;

                return popedElement;
        }
};

int main()
{
        int a[5] = { 5,9,2,7,3 };
        //display a in same order
        QUEUE<int>   q;
        //insert data of array a in a queue
        for (int i = 0; i < 5; ++i)
                q.pushQueue(a[i]);
        //display queue
        //use displayQueue, if you need to use the queue
        //again in this program
        cout << "the queue using displayQueue\n";
        q.displayQueue();
        //use popQueue if you are not going to use the queue
        //in this program
        cout << "the queue using popQueue\n";
        while (!q.emptyQueue())
        {
                int x = q.popQueue();
                cout << x <<"-->";
        }
        cout << "NULL\n";
        system("pause");
        return 0;
}
```

```
/*-----------------------------------output---------------------
the queue using displayQueue
5-->9-->2-->7-->3-->NULL
the queue using popQueue
5-->9-->2-->7-->3-->NULL
Press any key to continue . . .
------------------------------------------------------------------*/
```

c. Pointer implementation of ordered list

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

template <class T>
class ORDER
{
private:
	struct node
	{
		T	info;
		node *next;
	};
	node *order;
public:
	ORDER() { order = NULL; }// constructore
	bool emptyOrder()
	{
		return (order == NULL) ? true : false;
	}
	void pushOrder(T x)//
	{
		//insert x in the list and keep the list sorted
			node* r = new node; r->info = x;
			r->next = NULL;
			//find the insertion place;
			node* p = order; node* q = order;
			if (order == NULL)
				order = r;
			else
			{
				while (p != NULL && x > p->info)
				{
					q = p; p = p->next;
				}
				if (p == q)
				{  //insert in front
					r->next = p; order = r;
				}
				else
				{   //insert at the rear
					r->next = p; q->next = r;
				}

			}
		}


	void displayOrder()
```

```
	{
		node *p = order;
		while (p != NULL)
		{
			cout << p->info << "-->"; p = p->next;
		}
		cout << "NULL\n";
	}
	T popOrdere()
	{
		//return the info of the first node and then
		//delete that node
		T popedElement;
		node *p = order;
		popedElement = p->info;
		order = p->next;
		delete p;

		return popedElement;
	}
};

int main()
{
	int a[5] = { 5,9,2,7,3 };
	//display a in sorted form
	ORDER<int>   ord;
	for (int i = 0; i < 5; ++i)
		ord.pushOrder(a[i]);
	//display the list
	cout << "This is the ordered list\n";
	ord.displayOrder();

	//display the list using popOrder
	cout << "ordered list is \n";
	while (!ord.emptyOrder())
	{
		int x = ord.popOrdere();
		cout << x << "-->";
	}
	cout << "NULL\n";
	system("pause");
	return 0;
}
/*---------------------output--------------------------------
This is the ordered list
2-->3-->5-->7-->9-->NULL
ordered list is
2-->3-->5-->7-->9-->NULL
Press any key to continue . . .
-----------------------------------------------------------*/
```

Examples: Given two sets A={ 3,9,5, 7, 4} and B={4, 9, 8}. Find the intersection of A and B (the set of their common elements).

We use the class ORDER. This is how the main should look like

```
int main()
{
	//find the intersection of two sets A and B
	int A[5] = { 3,9,5,7,4 };
	int B[3] = { 4,9,8 };

	// insert elements of A in setA
	ORDER<int> setA; ORDER<int> setB;
	for (int i = 0; i < 5; ++i)
		setA.pushOrder(A[i]);

	//insert elements of B in setB
	for (int i = 0; i < 3; ++i)
		setB.pushOrder(B[i]);

	//display both sets
	cout << "setA = "; setA.displayOrder();
	cout << "setB = "; setB.displayOrder();

	//find AIB, the intersection of A and B
	ORDER<int> setAIB;
	int A_elt = setA.popOrder();
	int B_elt = setB.popOrder();
	while (!setA.emptyOrder() || !setB.emptyOrder())
	{
		if (A_elt == B_elt)
		{
			setAIB.pushOrder(A_elt);//collect their common elements
			A_elt = setA.popOrder();// go to next elt of setA
			B_elt = setB.popOrder();// and the next elt of setB
		}
		else if( A_elt < B_elt )
			A_elt = setA.popOrder();//look at the next element of setA
		else
			B_elt = setB.popOrder();//look at the next element of setB
	}
	if (A_elt == B_elt) setAIB.pushOrder(A_elt);

	//display their intersection set, setAIB
	cout << "setAIB = "; setAIB.displayOrder();

	system("pause");
	return 0;
}
/*---------------------output-------------------------
This is the ordered list
2-->3-->5-->7-->9-->NULL
ordered list is
2-->3-->5-->7-->9-->NULL
setA = 3-->4-->5-->7-->9-->NULL
setB = 4-->8-->9-->NULL
setAIB = 4-->9-->NULL
----------------------------------------------------*/
```

Data Structure

Project No.8 Name ……

<u>This is the last notice</u>. You are restricted to write your programs for all project's questions using your class knowledge. Using anything not covered in class are not acceptable. Thanks

1. Given two sets A={ 3 , 8, 4,1}, B={4,8,6,5, 7 }. Write a program to display their union (set of items which are either in A or in B or both with no duplicates). Use int A[4]={3,8,4,1}, B[5]={5,8,6,4,7}; and insert the elements of each set into an ordered linked list before finding their union

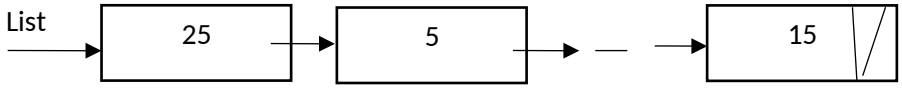
   Sample I/O
   Set A= 1→ 3 → 4→8 → NULL
   Set B= 4→5→6→7→8→NULL

   A union B = 1→3→4→5→6→7→8→NULL

2. Create the following text file. One way to sort the names in alphabetical order with O(n) is to insert the names in an ordered linked list. Write a program to insert the names in an ordered linked list. Display the linked list to see names in sorted form.

<u>Text file</u>
Trump_Donald
Obama_Barack
Bush_W_George
Clinton_Bill
Bush_H_George
Reagon_Ronald

3. Given the following linked list and declarations, write a few statements (not a complete program) to do the following. Please type your answers.




```
struct  node
{   int   info;
    node*   next;
};
node*  List;
```

Any pointer you want to use must be declared

Insert 10 in front of the list	Delete the last node

Insert 50 before the node whose info is 70	Insert 100 at the rear of the list
Delete the node whose info is 60	Determine the number of nodes in the linked list

```cpp
// Question_1.cpp : This file contains the 'main' function. Program execution begins and ends there.
// /
//------------------------------------------------------------------------------------------------
//Name                          Sai Chaitanya Kilambi
//Course                        CPSC 131 Data Structures, Fall, 2022
//Assignment                    No.8 question:1
//Due date                      10/26/2022
// Purpose:
// This program stores the data in ordered linked list and displays their union
//------------------------------------------------------------------------------------------------
// list of libraries
//
//importing the required libraries

#include <iostream>

using namespace std;

template <class T>
class ORDER
{
private:
    struct node
    {
        T info;
        node* next;
    };
    node* order;
public:
    ORDER() { order = NULL; }// constructore
    bool emptyOrder()
    {
        return (order == NULL) ? true : false;
    }
    void pushOrder(T x)//
    {
        //insert x in the list and keep the list sorted
        node* r = new node; r->info = x;
        r->next = NULL;
        //find the insertion place;
        node* p = order; node* q = order;
        if (order == NULL)
            order = r;
        else
        {
```

```
45              while (p != NULL && x > p->info)
46              {
47                  q = p; p = p->next;
48              }
49              if (p == q)
50              { //insert in front
51                  r->next = p; order = r;
52              }
53              else
54              { //insert at the rear
55                  r->next = p; q->next = r;
56              }
57          }
58      }
59      void displayOrder() {
60          node* p = order;
61          while (p != NULL)
62          {
63              cout << p->info << "-->"; p = p->next;
64          }
65          cout << "NULL\n";
66      }
67      T popOrder()
68      {
69          //return the info of the first node and then
70          //delete that node
71          T popedElement;
72          node* p = order;
73          popedElement = p->info;
74          order = p->next;
75          delete p;
76          return popedElement;
77      }
78  };
79
80
81  int main()
82  {
83      //find the union of two sets A and B
84      int A[4] = { 3,8,4,1 };
85      int B[5] = { 5,8,6,4,7 };
86      // insert elements of A in setA
87      ORDER<int> setA; ORDER<int> setB;
88      for (int i = 0; i < 4; ++i)
89          setA.pushOrder(A[i]);
90      //insert elements of B in setB
91      for (int i = 0; i < 5; ++i)
92          setB.pushOrder(B[i]);
93      //display both sets
```

```
 94      cout << "Set A = "; setA.displayOrder();
 95      cout << "Set B = "; setB.displayOrder();
 96      //find AB, the union of A and B
 97      ORDER<int> setAB;
 98      int A_elt = setA.popOrder();
 99      int B_elt = setB.popOrder();
100      while (!setA.emptyOrder() || !setB.emptyOrder())
101      {
102          if (A_elt == B_elt)
103          {
104              setAB.pushOrder(A_elt);//collect their common elements
105              A_elt = setA.popOrder();// go to next elt of setA
106          }
107          else {
108              if (A_elt < B_elt) {
109                  setAB.pushOrder(A_elt);
110                  A_elt = setA.popOrder();//look at the next element of setA
111
112              }
113              else {
114
115                  B_elt = setB.popOrder();//look at the next element of setB
116                  setAB.pushOrder(B_elt);
117              }
118          }
119      }
120
121      //display their union set, setAB
122      cout << "A union B = "; setAB.displayOrder();
123      return 0;
124  }
125
126
127
```

```cpp
// Question_2.cpp : This file contains the 'main' function. Program execution begins and ends there.
//// /
//------------------------------------------------------------------------------------------------
//Name                          Sai Chaitanya Kilambi
//Course                        CPSC 131 Data Structures, Fall, 2022
//Assignment                    No.8 question:2
//Due date                      10/26/2022
// Purpose:
// This program stores the data in ordered linked list from a text file and displays the data in order
//-------------------------------------------------------------------------------------------------
// list of libraries
//
//importing the required libraries


#include <iostream>
#include<string>
#include<fstream>

using namespace std;
template <class T>
class ORDER
{
private:
    struct node
    {
        T info;
        node* next;
    };
    node* order;
public:
    ORDER() { order = NULL; }// constructore
    bool emptyOrder()
    {
        return (order == NULL) ? true : false;
    }
    void pushOrder(T x)//
    {
        //insert x in the list and keep the list sorted
        node* r = new node; r->info = x;
        r->next = NULL;
        //find the insertion place;
        node* p = order; node* q = order;
        if (order == NULL)
            order = r;
```

```
45          else
46          {
47              while (p != NULL && x > p->info)
48              {
49                  q = p; p = p->next;
50              }
51              if (p == q)
52              { //insert in front
53                  r->next = p; order = r;
54              }
55              else
56              { //insert at the rear
57                  r->next = p; q->next = r;
58              }
59          }
60      }
61      void displayOrder()
62      {
63          node* p = order;
64          while (p != NULL)
65          {
66              cout << p->info << "-->"; p = p->next;
67          }
68          cout << "NULL\n";
69      }
70      T popOrdere()
71      {
72          //return the info of the first node and then
73          //delete that node
74          T popedElement;
75          node* p = order;
76          popedElement = p->info;
77          order = p->next;
78          delete p;
79          return popedElement;
80      }
81  };
82  int main()
83  {
84      string presidents[6];
85      std::fstream f;
86      //opening the file to only read
87      f.open("data.txt", std::ios::in);
88
89      // copying the file data into an array
90      for (int i = 0; i < 6; i++) {
91          f >> presidents[i];
92      }
93          //display a in sorted form
```

```
 94          ORDER<string> ord;
 95          for (int i = 0; i < 6; ++i)
 96              ord.pushOrder(presidents[i]);
 97          //display the list
 98          cout << "This is the ordered list\n";
 99          ord.displayOrder();
100  }
```