Data Structure	Week (4)

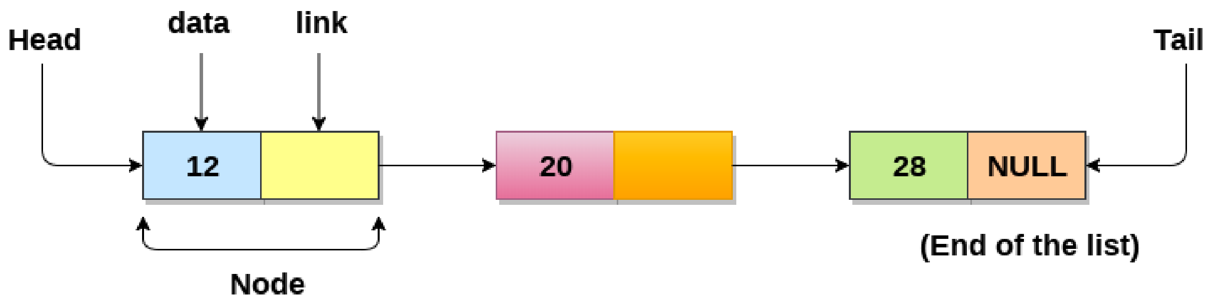
Section Content:

- Single Linked List

Linked List

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations.

- Dynamically Allocated Lists
- Size can vary at runtime
- No memory Wastage
- Elements can be inserted till memory is available



Implementation Node of a Single lined list

```
class Node
    {
        public int data;
        public Node next;
    }
```

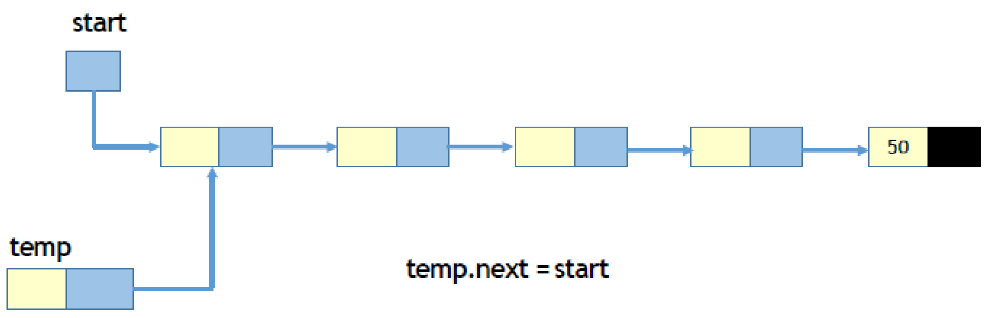
Single lined list

```
class SingleLinkedList
    {
        Node head;
```

Insertion in a Single Linked list

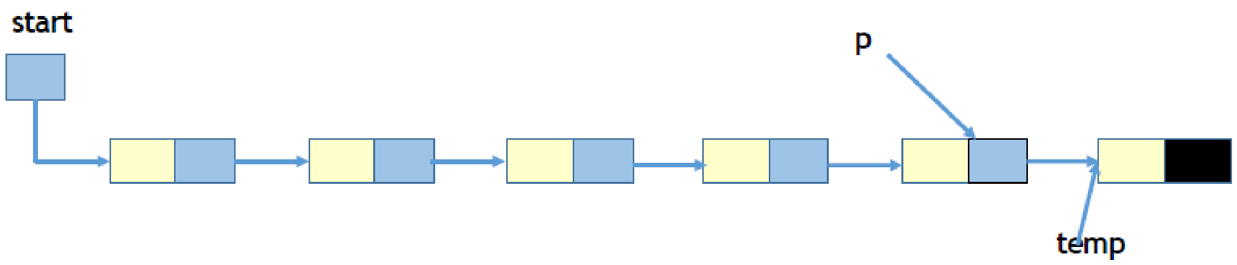
- Insertion in the beginning
- Insertion at the end
- Insertion in between the list nodes
- Insertion at a given position

Insertion in the beginning



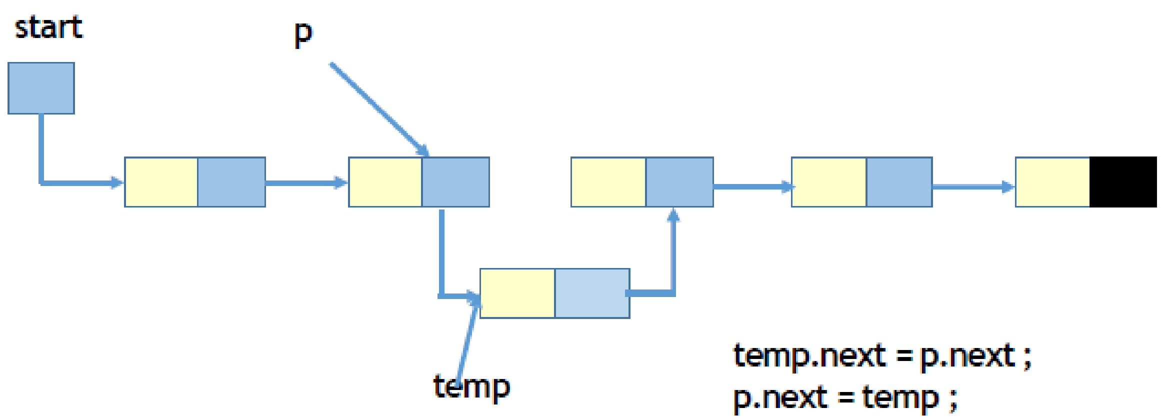
```
public void insertAtFront(int data)
        {
            Node temp = new Node();
            temp.data = data;
            temp.next = head;
            head = temp;
        }
```

Insertion at the end



```
public void insertAtEnd(int data)
        {
            Node temp = new Node();
            temp.data = data;
            if (head == null)
            {
                head = temp;
                return;
            }

            Node p = head;
            while (p.next != null)
                p = p.next;

            p.next = temp;
            temp.next = null;
        }
```

Insertion after a node

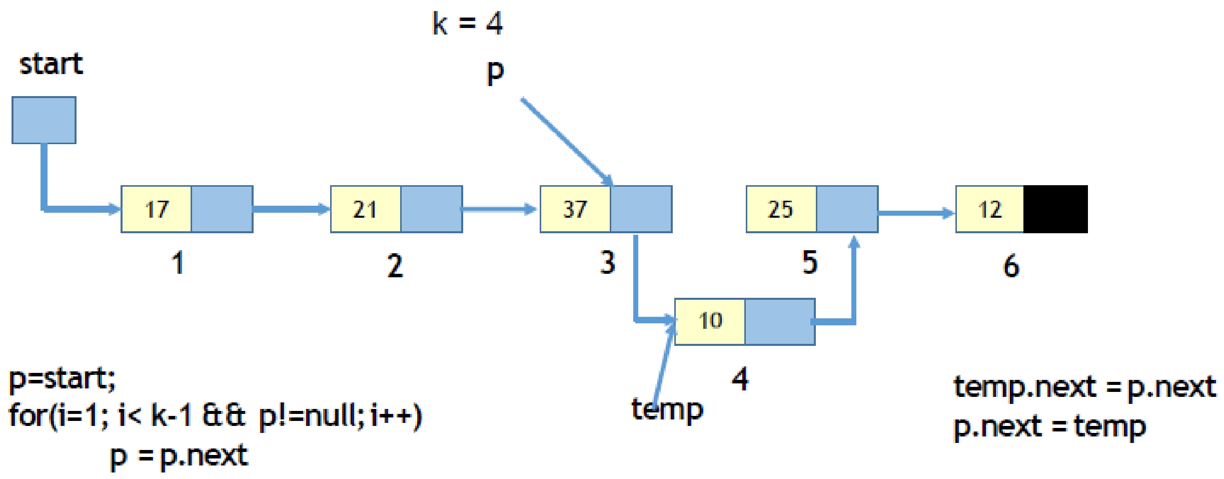


```
public void insertAfter(int data, int item)
        {
            Node temp = new Node();
            temp.data = data;
            Node p = head;
            while( p!= null)
            {
                if(p.data == item)
                {
                    temp.next = p.next;
                    p.next = temp;
                    return;
                }
                p = p.next;
            }
            Console.WriteLine("{0} not present in the list\n", item);
        }
```

Insertion before a node

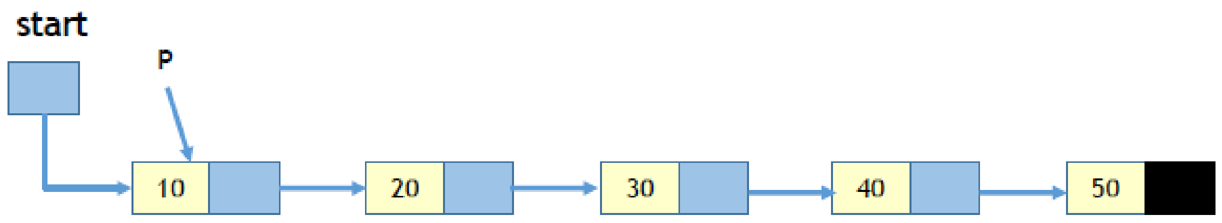
```
public void insertBefore(int data, int item)
        {
            Node temp = new Node();
            temp.data = data;
            if (head == null)
            {
                Console.WriteLine("List is empty!");
                return;
            }
            if(item == head.data)
            {
                temp.next = head;
                head = temp;
                return;
            }
            Node p = head;
            while(p != null)
            {
                if (p.next.data == item)
                {
                    temp.next = p.next;
                    p.next = temp;
                    return;
                }
                p = p.next;
            }
            Console.WriteLine("{0} not present in the list\n", item);

        }
```

Insertion at a given position



```
public void insertAtPosition(int data, int pos)
        {
            if (pos == 1)
            {
                insertAtFront(data);
                return;
            }
            Node temp = new Node();
            temp.data = data;
            Node p = head;
            for (int i = 1; i < pos - 1 && p != null; i++)
                p = p.next;
            if (p == null)
                Console.WriteLine("There are less than {} elements\n", pos);
            else
            {
                temp.next = p.next;
                p.next = temp;
            }
        }
```

Traversal of Linked list



```
public void printList()
        {
            Node p = head;
            while (p != null)
            {
                Console.Write(p.data + " ");
                p = p.next;
            }
           Console.WriteLine();

        }
```

Counting Nodes in a linked List

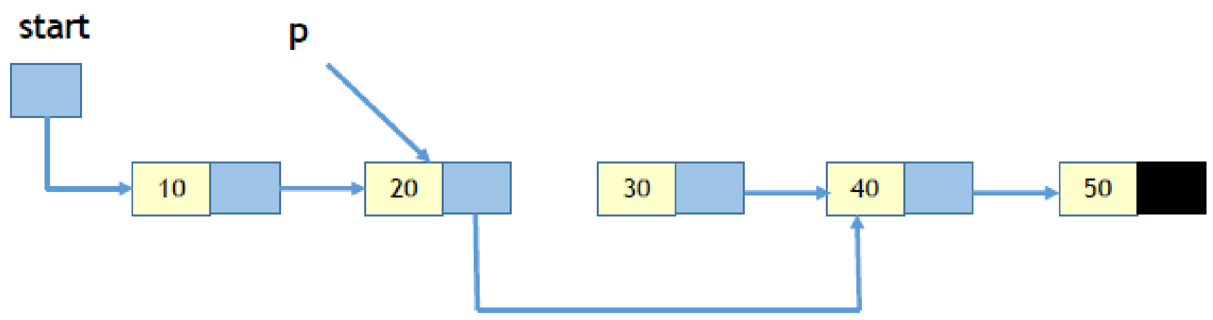
```
public int count()
        {
            int count = 0;
            Node p = head;
            while(p != null)
            {
                count++;
                p = p.next;
            }
            return count;

        }
```

Searching in a Linked List

```
public void search(int data)
        {
            Node p = head;
            int pos = 1;
            while(p != null)
            {
                if(p.data == data)
                {
                    Console.WriteLine("Item {0} found at position {1}", data,
pos);
```

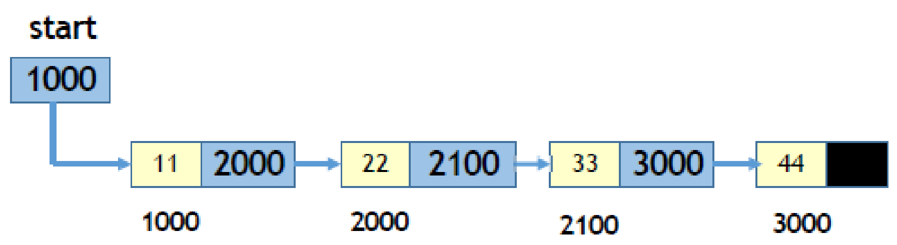
```
                    return;
                }
                p = p.next;
                pos++;
            }
            Console.WriteLine("Item {0} not found in list\n", data);

        }
```

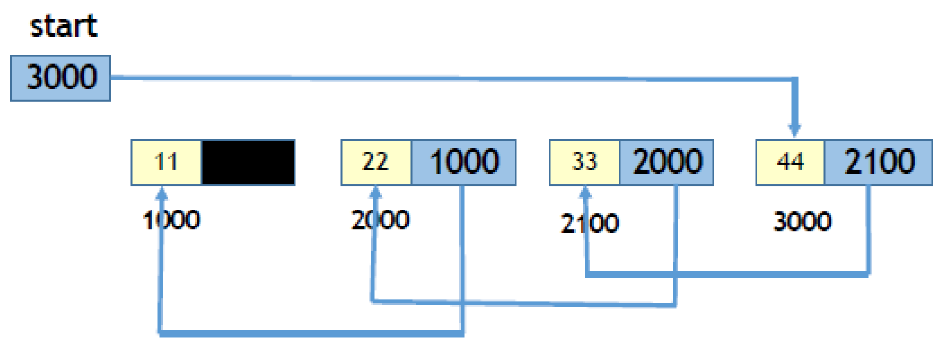
Deletion in a Single Linked List



```
public void deleteNode(int data)
        {
            if (head == null)
            {
                Console.WriteLine("List is empty");
                return;
            }
            if (head.data == data) //first node
            {
                head = head.next;
                return;
            }
            Node p = head;
            while (p.next != null)
            {
                if (p.next.data == data)
                {
                    p.next = p.next.next;
                    return;
                }
                p = p.next;
            }
            Console.WriteLine("Element {0} not found", data);
        }
```

Reversing a single linked List





```
public void reverseList()
        {
            Node prev, curr, next;
            prev = null;
            curr = head;
            while(curr != null)
            {
                next = curr.next;
                curr.next = prev;
                prev = curr;
                curr = next;
            }
            head = prev;

        }
```