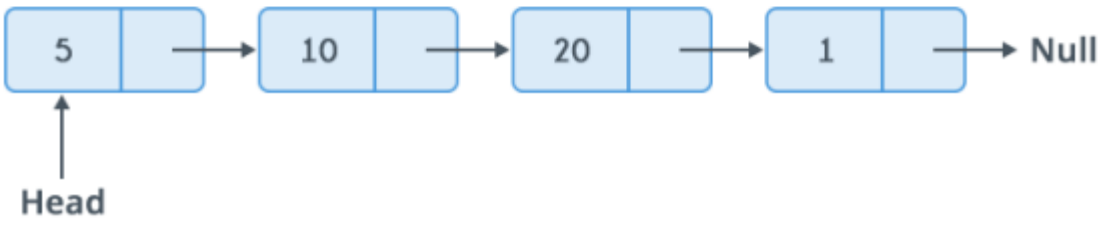
LAB 6
Linked List

In a game of Treasure Hunt, you start by looking for the first clue. When you find it, instead of having the treasure, it has the location of the next clue and so on. You keep following the clues until you get to the treasure.

A linked list is a way to store a collection of elements. Like an array these can be character or integers. Each element in a linked list is stored in the form of a node. A node is a collection of two sub-elements or parts. A data part that stores a number, a string or any other type of data and a next part that stores the link to the next node.

A linked list is formed when many such nodes are linked together to form a chain. Each node points to the next node present in the order. The first node is always used as a reference to traverse the list and is called HEAD. The last node points to NULL.

Pictorial Representation of Linked List:



Example: 6.1

```
class Node:
    def __init__(self, value):
        self.Info = value
        self.Next = None

    def Print(self):
        print(self.Info)
        if(self.Next is not None):
            self.Next.Print()

Start = Node(22)
Start.Next = Node(24)
Start.Next.Next = Node(25)
Start.Print()
```

Create Linked List with OOP Concept:

Example: 6.2 (Linked List Class)

```
class LinkedList:
    def __init__(self):
        self.Start = None

    def __init__(self, value):
        self.Start = Node(value)

    def Print(self):
        if(self.Start == None):
            print("List is Empty")
        else:
            self.Start.Print()
```

Insertion in Linked List

Example: 6.3 (Add InsertatBegin Method in Linked List Class)

```
class LinkedList:
    def __init__(self):
        self.Start = None

    def __init__(self, value):
        self.Start = Node(value)

    def InsertatBegin(self, value):
        if(self.Start == None):
            self.Start = Node(value)
        else:
            Temp = Node(value)
            Temp.Next = self.Start
              self.Start = Temp
```

Example: 6.4 (Add Insertatlast method in Linked List Class)

```
    def InsertatEnd(self, value):
        if (self.Start is None):
            self.Start = Node(value)
        else:
            ptr = self.Start
            while(ptr.Next != None):
                ptr = ptr.Next
            ptr.Next = Node(value)
```

Example: 6.5 (Add Count method in Linked List Class)

```
def Count(self):
    if (self.Start is None):
        print("List is Empty")
    else:
        ptr = self.Start
        count = 1
        while(ptr.Next != None):
            ptr = ptr.Next
            count += 1
    return count
```

Example: 6.6 (Add InsertonPosition method in Linked List Class)

```
def InsertonPosition(self, Position, value):
    if(Position == 1):
        self.InsertatBegin(value)
    elif (Position > 1 and Position <= self.Count() + 1):
        ptr = self.Start
        for i in range(1, Position - 1):
            ptr = ptr.Next
        New = Node(value)
        New.Next = ptr.Next
        ptr.Next = New
    else:
        print("Position Not Found in List")
```

Searching in Linked List:

Example: 6.7 (Add Search method in Linked List Class)

```
def Search(self, value):
  if(self.Start is None):
      print("List is Empty")
  else:
      ptr = self.Start
      Count = 1
      while(ptr != None and ptr.Info != value):
          ptr = ptr.Next
          Count += 1

      if(ptr == None):
          print(f"Value {value} Not found is List")
      else:
          print(f"Value {ptr.Info} found on Position {Count}")
```

Example: 6.8 (Add InsertAfter method in Linked List Class)

```
    def Search(self, value):
        if(self.Start is None):
            print("List is Empty")
        else:
            ptr = self.Start
            Count = 1
            while(ptr != None and ptr.Info != value):
                ptr = ptr.Next
                Count += 1
            if(ptr == None):
                return -1
            else:
                return Count

    def InsertAfter(self, Find, value):
        if(self.Start == None):
            print("list is Empty")
        elif(self.Search(Find) >= 0):
            self.InsertonPosition(self.Search(Find) + 1, value)
        else:
            print("Position Not Found")
```

Example: 6.9 (Add InsertBefore method in Linked List Class)

```
      def InsertBefore(self, Find, value):
        if(self.Start == None):
            print("list is Empty")
        elif(self.Search(Find) >= 0):
            self.InsertonPosition(self.Search(Find), value)
         else:
             print("Position Not Found")
```

Example: 6.10 (Add ChangeValue method in Linked List Class)

```
      def ChangeValue(self, Find, value):
          if (self.Start == None):
              print("List is Empty")
          else:
              ptr = self.Start
              while(ptr != None and ptr.Info != Find):
                  ptr = ptr.Next

              if(ptr == None):
                  print(f"Value {Find} not found in List!")
              else:
                 ptr.Info = value
                 print(f"Value {Find} has been changed with {ptr.Info}")
```

Deletion in Linked List:

Example: 6.11 (Add DeleteFormEnd method in Linked List Class)

```
def DeleteFromEnd(self):
    if (self.Start is None):
        print("List is Empty")
    elif (self.Start.Next is None):
        self.Start = None
    else:
        ptr = self.Start;
        while(ptr.Next.Next is not None):
            ptr = ptr.Next
        ptr.Next = None
```

Example: 6.12 (Add DeleteFormBegin method in Linked List Class)

```
def DeleteFromBegin(self):
    if(self.Start is None):
        print("List is Already Empty")
    else:
        self.Start = self.Start.Next
```

Example: 6.13 (Add DeleteFormPosition method in Linked List Class)

```
def DeleteFromPosition(self, Position):
        if(self.Start == None):
            print("List is Already Empty")
        elif(Position == 1):
            self.Start = self.Start.Next
        elif(Position > 0 and Position <= self.Count()):
            ptr = self.Start
            for i in range(1, Position - 1):
                ptr = ptr.Next
            ptr.Next = ptr.Next.Next
        else:
            print("Position Not Find")
```

Lab 6 Exercises:

1. Write python code for Deletion of node from the head using linked list.

2. Create a Linked list and insert following values 50, 30, 8, 65, 89, 85, 7 respectively. Apply any sorting algorithm on this list to sort in Ascending Order. Print Sorted Linked List in the end.