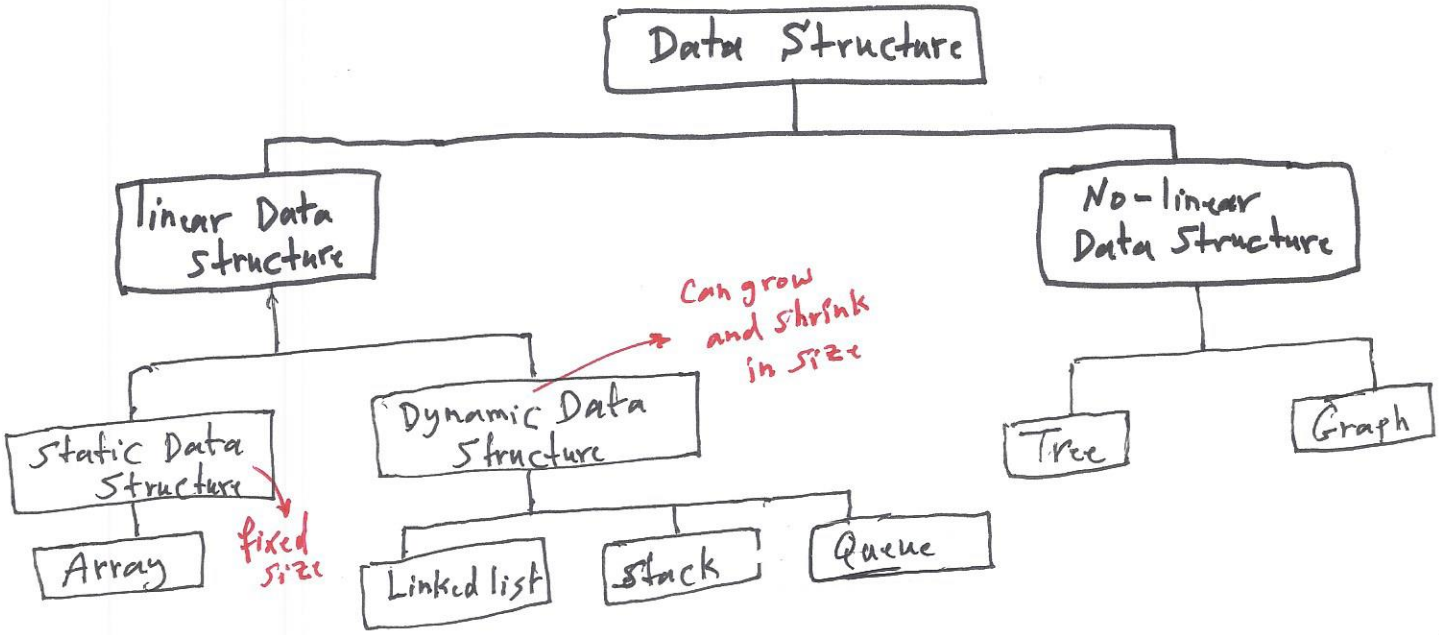
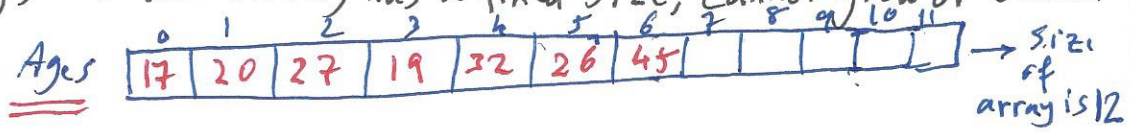
* Test #1, Sat. 3-9-24, OOP, inheritance, pointers, virtual funcs., static funcs, operatorOverloading. (17 Sp.24 Sat.)
  - Multiple choice and T/F Questions, ScanTron 882-E
  You can find review questions under week 4 in Canvas.

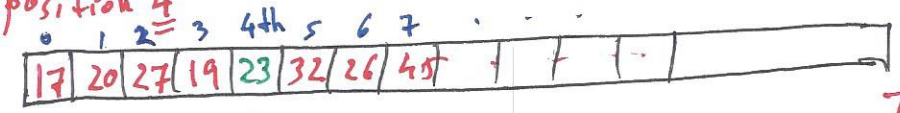
Week 5, Sat.: Classification of Data Structure



We have used arrays so far. Array has a fixed size, cannot grow or shrink in size.



e.g. Insert 23 at position 4



O(n) Time Complexity

Linked List

Advantages:
1- Use as much as memory you need
2- Easy to add and delete data
3- It is dynamic data structure. (grow or shrink)

Disadvantages:
1- Random access is not allowed.
2- Extra memory for a pointer is required
3- Not Cache-friendly
4- Reverse traversing is not possible
5- Searching for an element (or Node) is Costly and need time, Time Complexity is O(n)
6- Sorting of linked list is very Complex and Costly and we need

time, Time Complexity is O(n).

To Create a Node: in C++, we can use struct or class.

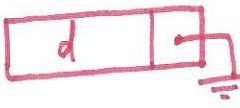
Using struct:

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

The structure that makes it possible to create nodes that points to other nodes of the same type.

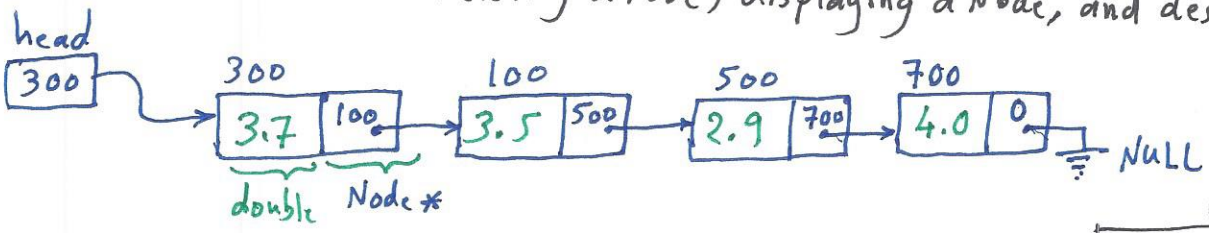
Using Class: We can use a class to create a Node:

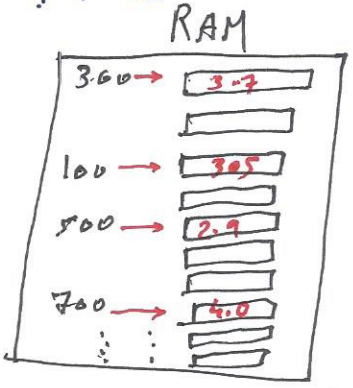
```
class Node
{
  public:
     int data;
     Node *next;
  public Node (int d) // Constructor with one argument
  {  data=d;
     next=NULL;
  }
};//end class
```



nullptr has address 0
It doesn't point to anything.

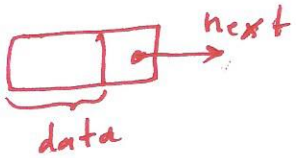
Linked List Operations: e.g. Adding a Node, traversing a list, inserting a Node, deleting a Node, displaying a Node, and destroying a list.



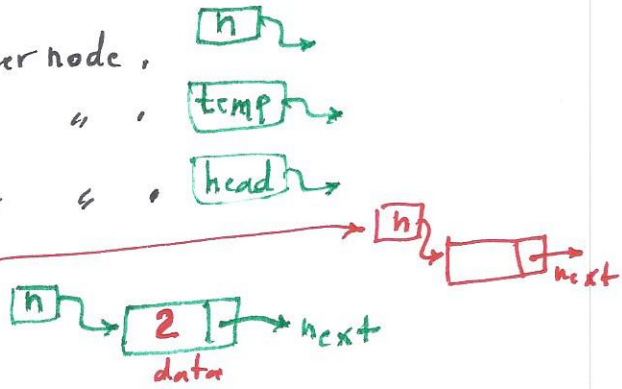


- Create a linked List:

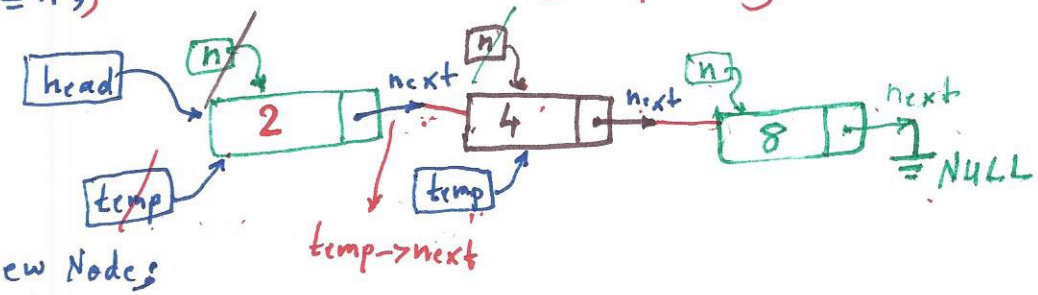
```
struct Node
{  int data;
   Node *next;
};
```




```
Node *n ; // Node pointer which points to another node.
Node *temp; // " " " " " " " " " .
Node *head; // " " " " " " " " " .
n = new Node ; // or Node *n = new Node;
n->data = 2 ; // n dot data
```




```
temp = n ;
head = n ;
```
→ point to whatever n is pointing to.

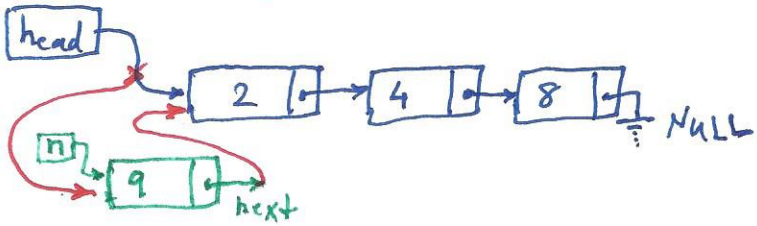



```
n = new Node;
n->data = 4;
temp->next = n; // Connects 1st node to 2nd Node
temp = n;
```

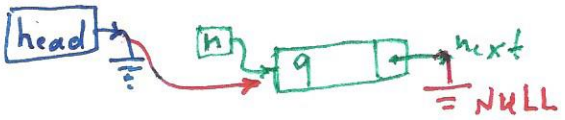
```
n = new Node ;
n->data = 8;
temp->next = n ; // Connects 2nd Node to the 3rd Node
n->next = NULL ; //address 0
```

- Adding a Node at the beginning of a linked list:

```
void addNode ( int num )
{  Node *n = new Node ; // Dynamic Memory Allocation
   n->data = num; // 9
   if ( head != NULL )
   {  n->next = head;
      head = n;
   }
   else // empty list
```



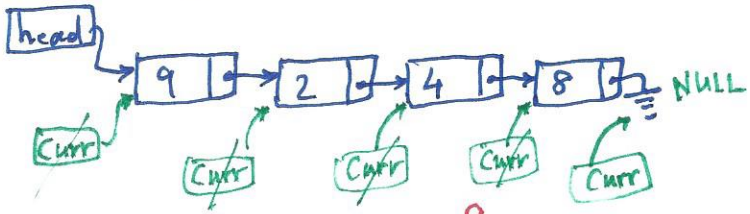
```
{ head=n;
  n->next=NULL;
}
```




```
}//end addNode
```

To print the Contents of a linked list (traversing a linked list):

```
Void display()
{ Node *Curr;
  Curr=head;
  While (Curr!=NULL)
  { Cout << Curr->data <<endl;
    Curr=Curr->next;
  }
}//end display
```



9
2
4
8