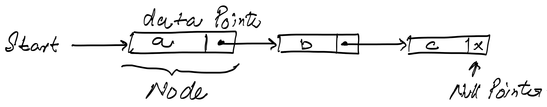
Linked List

Sunday, 20 February 2022 5:18 PM



Keywords:

- Node
- Pointer
- Null Pointer.
- Start Pointer.

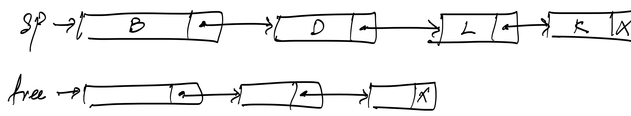
Rules:

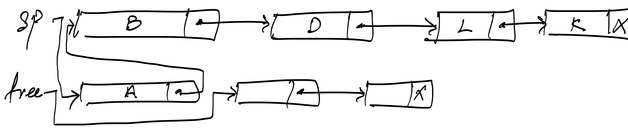
- All elements, node data, should be inserted in Ascending order.
- It is one-way (forward only) LL.
- We only move forward with pointer.
- No. of nodes is unknown.
- Whenever a new node is taken, it is taken from the start of free list.
- When a node is deleted, it is assigned to the start of free list.

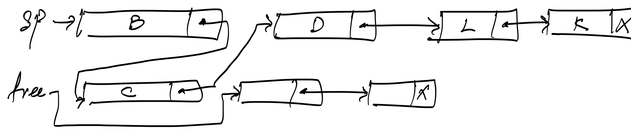
Types:

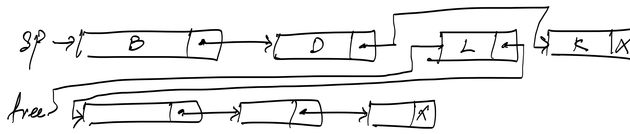
- Ordered LL.
- Unordered LL.

Conceptual working (Structured english).



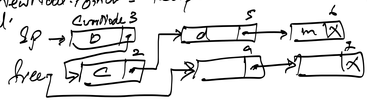






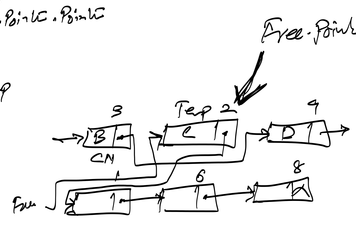
// Inserting a node.

1. INPUT data; d.
2. IF free = Null Then
   display error
   end
3. grab a node from free list; newNode.
4. Assign value; NewNode.data = d.
5. IF sp = Null Then
   sp = NewNode
   NewNode.pointer = Null
   End.
6. Reach to the node after which NewNode is inserted; CurrNode
7. Temp = CurrNode.Pointer
8. CurrNode.Pointer = NewNode
9. NewNode.Pointer = Temp.
10. End.



// delete a node from LL.

1. IF SP = Null Then
   show error
   End.
2. Reach to the node before the node to be deleted.
   call it CN.
3. Temp = CN.Pointer
4. CN.Pointer = CN.Pointer.Pointer
5. FTemp = Free
6. Free = Temp
7. Free.Pointer = FTemp



Free = 1 Temp = 2

Free.Pointer = 6

FTemp = 1

// Traverse / Search

0. INPUT Value.
1. GOTO SP; Call it CN → isfound = False
2. WHILE CN.Pointer <> Null IF CN.data = Value Then
3. OUTPUT CN.data → isfound = True
4. GOTO CN.Pointer; Call it CN
5. END WHILE
6. If isfound = False Then output "Value not found!!!"