AVIRATH MANOHAR HEGDE
1BM19CS195

25-11-2020

LAB-6 → SINGLE LINKED LIST IMPLEMENTATION

```
A. Struct NODE
 {
   int data
   Struct NODE * next
 }

Push-front ( struct NODE * )* head , int new-data )
{
   Struct Node* new-node = ( struct Node* )malloc ( size of ( Struct Node))
   new_node → data = new_data
  new_node → next = *head
  * head  = new_node
 }

Push_end ( Struct Node ** head , int new-~~data~~)
 {
  Struct Node * new_node = ( struct Node * ) malloc ( size of Node))
  Struct Node* Last = *head
 new_node → data = new-data
 new-node → next = NULL
 if (*head == NULL)
 {
    *head = new_node
    return;
 }

while ( Last → next != NULL)
    Last = Last → next
 Last → next = new_node
  return;
}
```

```
Push_specifiepos (int data, int Position)
 {
   Struct NODE * new_node = (Struct Node*) malloc (sizeof (Struct Node))
   new_node -> data = data
   int i;
   Struct Node *temp = head
   if (position == 1)
 {
   new_node -> next = temp
       head = new_node
     return
 }
 for (i=1; i<position-1; i++)
 {
 temp = temp->next
  new_node -> next = temp->next
   temp->next = new_node
 }

Print_LinkedList (struct Node *node)
 {
   while (node! = NULL)
    {
      Printf ("%d", node->data)
    node = node -> next
    }
 }
```

```
delet _front()
{
  Struct Node* ptr;
  if (head = = NULL
  {
   Printf ("list is Empty")
  }
 else
 {
   Ptr= head
     head = ptr -> next
   free(ptr)
 }
}

delete _end()
 {
   Struct Node* ptr, *ptr1,
   if (head = = NULL)
     List is Empty!!
else if(head -> next = =NULL)
{
   head =NULL
   free(head)
   }
 else
{
 Ptr = head
   while (ptr -> next! = NULL)
  {
  ptr1 = ptr
  Ptr = ptr -> next
    }
ptr1 -> next = NULL
    free (ptr)
    }
}


delete_specific pos()
    {
    struct Node *ptr, *ptr1
    int  i, Position
   scanf ("%d", & position)
    ptr = head
 for (i=0; i<Position; i++)
    {
       ptr1 = ptr
       ptr = ptr -> next
       if (ptr == NULL )
       {
          "  less  than  required element in the list is
           return;
       }
    ptr1 -> next = ptr -> next
      free (ptr)
   }.
```