SPLASH

VINEETH
1BM19CS183

Program to implement singly linked list.

```
// for insertion
void insert_at_beginning()
{ struct node * ptr
  ptr→data = new_item
  ptr→next = head
  head = ptr
  Print "node inserted at beginning"
}

2 insert_at_last()
{ struct node *ptr, *Temp
  ptr=(struct node *) malloc (sizeof(struct node)) }
 ptr→data = new_item
  if(head==NULL)
  { ~~head~~ ptr→next = NULL
       head = ptr
  Print " node inserted ~~at last~~ "
 } }
else
   { temp = head
     while(temp→next != NULL)
     { temp ~~→~~ = temp→next }
     temp→next= ptr
     ptr→next = NULL
    Print "node inserted at last"
   }
 }
```

```
insert_at_pos ( )
{ struct node * ptr , *temp
   ptr -> data = new_elem
   temp = head
   if ( pos = = 1)
   { ~~head = ptr~~
        ptr -> next = h. temp
        head = ptr
     }    return
        }

 for (i=1 ; i< pos-1; i++)
  {  temp = temp -> next }
   ptr -> next = temp -> next
     temp -> next = ptr
 }

  // for deletion
 delete_at_begining ( )
{ struct node *ptr
   if (head == NULL)
  Print " List is Empty"
    ~~return~~
 else
    {  ptr = head
        head = ptr -> next
        free (ptr)
     Print " Node deleted from begining "
    }
 }
```

VINEETHA
1BM19CS183

```
delete_at_end ()
{ struct node * ptr, *ptr1
  if (head == NULL)
  Print "List is empty"
 else if (head -> next == NULL)
    { head = NULL
       free (head)
     Print " node is deleted"
    }
else
   { ptr = head
       while (ptr -> next ! = NULL)
        { ptr1 = ptr
          ptr = ptr -> next }
       ptr1 -> next = NULL
        free (ptr)
      Print " node deleted from last"
    }
  }


delete_specified_data ()
{ struct node * ptr, *ptr1
   ptr = head
 while ( ptr != NULL && ptr -> data != item )
   { ptr1 = ptr
       ptr = ptr -> next }
       ~~ptr~~ Print ptr -> data
          ptr1 -> next = NULL
          free (ptr)
      Print " is deleted from the list"
   }
```

Date __/__/__
Page ______
SPLASH

```
display ( )
{ struct node * temp
    temp = head
    if ( head = = NULL)
   Print "List is empty"
else
   {  while ( temp -> next ! = NULL)
       { Print  temp -> data
            temp -> temp -> next }
     }
}
```