SUSHMITHA Y.V
1BM19CS165
D-1

LAB 5 & LAB 6

Linked list implementation

```
// for insertion
  void insert_at_beginning()
{
  Struct node *ptr
  ptr -> data = new_item
  ptr -> next = head
   head = ptr
   print "node inserted at beginning"
}
  insert_at_last()
{
  Struct node *ptr, *temp
  ptr = (struct node *) malloc (size of (structure))
  ptr -> data = new_item
     if (head == NULL)
{
   ptr -> next = NULL
     head = ptr
  print "node inserted"
}
```

```
else
{
  temp = head
  while (temp -> next != NULL)
  { temp = temp -> next }
    temp -> next = ptr
    ptr -> next = NULL
     printf "node inserted at last"
  }
}
  insert_at_pos ()
{ struct node *ptr, * temp
    ptr -> data = new_item
     temp = head
     if (pos == 1)
{
  ptr -> next = 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_beginning()
    {
      struct node *ptr
      if (head == NULL)
      print " list is empty"
      else
        { ptr = head
          head = ptr -> next
          free (ptr)
          print " node deleted from beginning"
        }
    }
      delete_at_end()

{ struct node *ptr, *ptr1
      if (head == NULL)
```

```
printf "list is empty"

else if (head -> next == NULL)

    {
        head = NULL
        free (head)
        printf "node is deleted"
    }

else
    { ptr = head
      while (ptr -> next != NULL)
    {
        ptr1 = ptr
        ptr2 = ptr -> next
        ptr1 -> next = NULL
        free (ptr)
        printf "node deleted from last"
    }
}

delete_specified_data()

{
    Struct node *ptr, *ptr1
    ptr = head
    while (ptr != NULL && ptr -> data != item)
```

```
{
  ptr1 = ptr
  ptr = ptr -> next;
  printf ptr -> data
  ptr1 -> next = NULL
  free (ptr)
  printf " is deleted from the list"
}
display()
{
  struct node *temp
  temp = head
  if (head == NULL)
  printf " list is empty"
else
{
  while (temp -> next != NULL)
  {
    printf temp -> data
    temp -> temp -> next;
  }
}
```