AVIRATH MANOHAR HEGDE
1BM19CS195

14-12-2020

LAB - 9

→ Double Linked list implementation.

A.
```
Struct node
{
  Struct node *prev;
  int data;
  Struct node * next;
}

void delete(){
Struct node *ptr, struct node * temp;
int val;
Printf("Enter the value : ");
Scanf("%d", &val);
temp = head
while(temp → data != val)
  temp = temp → next
if (temp → temp→ next != NULL)
  Printf(" cannot be deleted");
else if (temp → next → next == NULL)
  temp → next = NULL;
  Printf(" last Node Deleted");
else
{
ptr = temp → next;
temp → next = ptr → next;
ptr → next → prev = temp;
free(ptr);
Printf(" Node Deleted ");
  }
}
```

```
void display() {
    struct node *ptr = head;
  if(ptr==head)
    printf("Empty list");
    else
     {
     printf("\n\n LIST →");
     while(ptr != NULL)
  {
    printf("%d", ptr → data);
      ptr = ptr → next;
    }
  }
}

void create_list(){
     struct node* ptr;
  int i, n, new data;
 printf("Enter the no. of nodes");
  scanf("%d", &n);
  if (n >= 1)
 {
   head = (struct node *) malloc (sizeof (struct node));
    if (head != NULL)
   {
   printf("Enter value of for Node 1: ");
   scanf("%d", &new-data);

   }
```

```
void insert(){
  int i, position, new_data;
  struct node *ptr, *temp;
  if(head == NULL)
{
  Printf("list is Empty");
  }
else
{
   temp = head; i=1;
   while(i<position-1 && temp != NULL)
   {
    temp = temp -> next;
     i++;
   }
if(position == 1)
{
  ptr -> data = new_data;
  ptr -> next = head;
  ptr -> prev = NULL;
  head -> prev = ptr;
head = ptr;
}
else if(temp == last)
  {
     ptr -> data = new_data;
     ptr -> next = NULL;
  ptr -> prev = last;
   last -> next = ptr;
   last = ptr;
   }
```

```
else if( temp != NULL
    {
    ptr -> data = new_data;
    ptr -> next = temp -> next;
      ptr -> prev = temp;
      if ( temp -> next != NULL)
   {
     temp -> next -> prev = ptr;
     }
   temp -> next = ptr;
   }
else
  {
    printf(" Invalid pos ");
        }
    }
}
```