PAGE No.
DATE / /

DS-LAB PROGRAM-6

Q. WAP to implement Singly linked list with operations:-
a) Create a linked list.
b) Deletion of first element, specified and last element in the list.
c) Display contents of the linked list.

Ans:
```
#include<stdio.h>
#include<conio.h>
struct node
  {
  int info
 struct node *link;
  };
typedef struct node *NODE
NODE getnode()
 {
 NODE x;
  x = (NODE)malloc(sizeof(struct node));
 if(x==NULL)
  {
   printf("mem full\n");
   exit(0);
  }
return x;
 }
void freenode(NODE x)
{
free(x);
 }
NODE insert_front(NODE first, int item)
```

Scanned with CamScanner

```
{
NODE temp;
 temp=getnode();
  temp->info=item;
  temp->link=NULL;
  if(first==NULL)
   return temp;
 temp->link=first;
 first=temp;
 return first;
 }
NODE delete_front(NODE first)
 {
  NODE temp;
  if (first==NULL)
   {
   printf("List is empty cannot delete\n");
   return first;
  }
 temp=first;
 temp=temp->link;
 printf("item deleted at front end is %d\n",first->info);
 free(first);
 ~~fre~~ return temp;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
 temp=getnode();
 temp->info =item;
 temp->link=NULL;
 if(first ==NULL)
```

PAGE No.
DATE / /

```
return temp;
cur=first;
while (cur->link!=NULL)
  {
   cur=cur->link;
   cur->link=temp;
   return first;
  }
NODE delete_rear (NODE first)
{ NODE cur,prev;
  if (first==NULL)
   {
    printf("List is empty cannot delete\n");
     return first;
   }
  if (first->link==NULL)
   {
    printf("item deleted is %d\n",first->
                                        info);

    free(first);
    return NULL;
   }
  prev=NULL;
 cur=first;
  while (cur->link!=NULL)
   {
    prev=cur;
    cur=~~first~~cur->link;
   } ~~while~~
  printf("item deleted at rear end is %d",
     cur->info);
  free (cur);
```

Scanned with CamScanner

DATE / /

```
prev->link=NULL;
 return first;
 }
NODE delete-pos(int pos, NODE first)
 {
  NODE prev, cur;
  int count;
  if (first ==NULL||pos<=0)
   {
    printf("invalid position\n");
    return NULL)
   }
  if(pos==1)
   {
    cur=first;
   first=first->link;
  printf("item deleted is %d", cur->info);
   freenode(cur);
   return first;
  }
 prev=NULL;
 cur=first;
 count=1;
while (cur!=NULL)
 {
 if(count==pos)
  {
   break;
  }
 prev=cur;
 cur=cur->link;
 count++;
}
```

PAGE No.
DATE / /

```
if(count!=pos)
 {
  printf("invalid position\n");
  return first;
 }
 pev->link = cur->link;
 printf("item deleted ~~a~~ is %d", cur->info);
 freenode(cur);
 return first;
}
void display(NODE first)
 {
  NODE temp;
  if(first==NULL)
   {
    printf("list empty cannot display items");
   }
  else
   {
   printf("contents of the list\n");
   }
 for(temp=first; temp!=NULL; temp=temp->link)

   {
    printf("%d\n", temp->info);
   }
 }
 void main()
{int item, choice, pos;
 NODE first=NULL;
 for(;;)
 {
```

PAGE No.
DATE / /

```
printf("\n1: Insert_front \2. Delete
 front-\n 3: Insert_rear 4: Delete_rear\5:
 Delete_pos \n 6. Display_list 7: Exit\n");
printf("Enter choice");
 scanf("%d",&choice);
 switch(choice);
 {
 case 1: printf("Enter item at front end");
        scanf("%d",&item);
       first= insert_front(item,first);
        break;
 case2:
      first=delete_front(first);
       break;
 case 3: printf("Enter item at rear end");
      scanf("%d",&item);
      first =insert_rear(first,item);
    break;
case 4: first=delete_rear(first);
      break;
case5:
     printf("Enter position:\n");
     scanf("%d",&pos);
     first=delete_pos(pos,first);
    break;
case6: display(first);
      break;
case7: exit(0);
    break;
  default:printf("Invalid choice\n");
  }
}
}
```