Date
Page No.

DS - Lab

Singly linked list

```
#include <stdio.h>
#include <stdlib.h>
 struct node
{
int info;
 struct node *link;
};
type def struct node* NODE;
  NODE get node()
   {
   NODE x;
x=(NODE) malloc (size of (struct node));
 if (x==NULL)
{
printf ("mem full\n");
   exit(0);
}
return x;
}
void free node (NODE x)
 {
 free(x);
}
NODE insert- front (NODE first, int item)
{
NODE temp;
temp = get node ();
temp → info = item;
temp → link = NULL;
```

```
if (first == NULL)
return temp;
temp -> link = first;
first = temp;
return temfirst;
}
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);
}
NODE insert_rear (NODE first, int item)
{
NODE temp, cur;
temp = getnode ();
temp -> info = item;
temp -> link = NULL;
if (first == NULL)
return temp;
cur = first;
while (cur -> link != NULL)
cur = cur -> link;
cur -> link = temp;
```

```
return first;
}
NODE cur, prev;
if (first == NULL)
{
printf("list is empty cannot delete \n");
 return first;
}
if (first -> link == NULL)
{
printf("list item deleted is %d \n", first -> info);
free (first);
return NULL;
}
prev = NULL;
cur = first;
while (cur -> link != NULL)
{
prev = cur;
cur = cur -> link;
}
printf("item deleted at ~~rear~~ rear end is %d", cur ->
                                                  info);
free (cur);
prev -> link = NULL;
 return first;
}
NODE insert_posth (int item, int pos, NODE first)
{
NODE temp;
NODE prev, cur;
 int count;
```

```
temp = getnode ();
temp -> info = item;
temp -> link = NULL;
if first == NULL && pos == 1)
  return temp;
  if (first == NULL)
{
printf (" invalid pos \n");
  return first;
  }
  if (pos == 1)
}
temp -> link = first;
  return temp;
}
count = 1;
prev = NULL;
cur = first;
while ( cur != NULL && count != pos)
{
prev = cur;
cur = cur -> link;
count ++
}
if (count == pos)
{
prev -> link = temp;
temp -> link = cur;
}return first;
}
printf (" IP \n");
  return first;
```

```
}
NODE delete_pos (int pos, NODE first)
{
if (first == NULL {
printf ("list empty \n");
return first;
}
NODE temp = first;
if (pos==1)
{
first = temp -> link;
free (temp);
return first;
}
NODE prev;
for (int i=1; temp != NULL && i<pos ; i++)
{
prev = temp;
temp = temp -> link;
}
if (temp == NULL || temp -> link == NULL){
printf ("Invalid posth \n");
return null NULL
}
void display (NODE first
{
NODE temp;
if (first == NULL)
printf (" list empty cannot display items \n");
for (temp = first ; temp! = NULL; temp = temp -> link)
{
printf ("%d \n", temp -> info);
```

```
}
}
NODE concat (NODE first, NODE second)
{
 NODE cur;
 if (first = NULL)
  return second;
  if (second = = NULL)
   return first;
   cur = first;
   while (cur→link ! =NULL)
 }
void main()
{
 int item, choice, pos, i, n;
  NODE a, b;
  NODE first = NULL;
for(;;)
{
printf(" 1. insert_front \n2 .delete_front\n3 insert
_rear \n4. delete_rear\n5. insert at pos\n6.
delete at pos \n7. concat\n8. reverse\n9.order
 list \n10. display \n");
 printf(" Enter the choice\n");
  scanf("%d", &choice);
   switch (choice)
  {
case 1: printf(" enter the item at front-end \n");
scan f ("%d", &item);
 first = insert_front (first, item);
  break;
```

```
case 2: first = delete_front (first);
 break;
case 3: printf ("enter the item at rear end \n");
scanf ("%d", &item);
first = insert_rear (first, item);
 break;
case 4: first = delete_rear (first);
 break;
case 5: printf ("Enter item \n");
 scanf ("%d", &item);
printf ("enter the post^n \n");
  scanf ("%d", &pos);
 first = insert_pos (item, pos, first);
 break;
 case 6:
 printf ("Enter post^m of deletion \n");
  scanf ("%d" &pos);
  first = delete_pos (pos, first);
 break;
 case 7:
 printf ("Enter the no. of nodes in 2 \n");
   scanf ("%d" &n);
    a = NULL;
    for (i=0; i<n; i++)
   {
   printf ("Enter the item \n");
    scanf ("%d" &item);
    ab = insert_rear (b, item);
   }
   a = concat (a, b);
    display (a);
    break;
```

```
case 9:
first = order-list (first);
break;
case 10: display (first);
break;
default : exit (0);
break;
}
}
}
```