Lab-Program-7

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.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)
{
 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);
   return temp;
}
```

```
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 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 = cur -> link;
}
Printf("Item deleted at rear-end is %d", cur->info);
free(cur);
Prev -> link = NULL;
return first;
}

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

```
    prev = cur;
    cur = cur -> link;
  }
  prev -> link = temp;
  temp -> link = cur;
  return first;
}

void display(NODE first)
{
  NODE temp;
  if (first == NULL)
  printf("List empty cannot display items\n");
  printf("contents of the list:\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)
  cur = cur -> link;
  cur -> link = second;
```

```
    return first;
}
NODE reverse(NODE first)
{
  NODE cur, temp;
  cur = NULL;
  while(first != NULL)
  {
    temp = first;
    temp = first -> link;
    temp -> link = cur;
    cur = temp;
  }
  return cur;
}
  int main()
{
  int item, choice, n, i;
  NODE first = NULL, a, b;
  for(;;)
  {
  Printf("\n1: Insert_front\n2: Delete_front
          \n3: Insert_rear \n4: Delete_rear\n");
  Printf("5: order_list\n6: Display_list\n7: concat
          \n8: Reverse \n9: Exit\n");
  Printf("Enter the choice :");
  Scanf("%d", &choice);
```

```
switch(choice)
{
case 1: Printf("Enter the item at front-end\n");
        scanf("%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 the item to be inserted
                in ordered_list\n");
        scanf("%d", &item);
        first= order_list(item, first);
        break;

case 6: display(first);

case 7: Printf("Enter the no: of nodes in list\n");
        scanf("%d", &n);
        a=NULL;
        for(i=0; i<n; i++)
        {
         Printf("Enter the item\n");
```

```
        scanf("%d", &item);
        a = insert_rear(a, item);
    }
    Printf("Enter the no of nodes in 2\n");
    scanf("%d", &n);
    b = NULL;
    for(i=0; i<n; i++)
    {
    Printf("Enter the item\n");
    scanf("%d", &item);
    b=insert_rear (b, item);
    }
    a = concat(a, b);
    display (a);
    break;
case 8 : first = reverse (first);
    display (first);
    break;
case 9 : exit(0);
    break;
default : Printf("Invalid choice\n");
}
}
}
```