Lab-Program - 5 and 6

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.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("Memory Full\n");
    exit(0);
  }
  return x;
}
void freenode(NODE x)
{
  free(x);
}
```

```
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 insert_Pos (int item, int Pos, NODE first)
{
  NODE temP, cur, Prev;
  int count;
  temP = getnode();
  temp->info = item;
  temP->link = NULL;
  if (first == NULL && Pos == 1)
  {
    return temp;
  }
```

```
if (first==NOLL)
{
  Printf("Invalid Position\n");
  return first;
 }
  if (Pos==1)
  {
    temp->link=first;
    first=temp;
    return temp;
   }
    count=1;
    Prev=NOLL;
    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("Invalid Position\n");
      return first;
}
```

```
NODE delete_Pos(int Pos, NODE first)
{
   NODE cur;
   NODE Prev;
   int count, flag=0;
   if(first==NULL || Pos<0)
   {
    Printf("Invalid Position\n");
    return NULL;
   }
    if(Pos==1)
    {
     cur=first;
     first=first->link;
      freenode(cur);
      return first;
    }
     Prev=NULL;
     cur=first;
     count=1;
     while(cur!=NULL)
     {
       if(count==Pos){flag=1; break;}
       count++;
       Prev=cur;
       cur=cur->link;
     }
```

```
if (flag==0)
{
 Printf("Invalid Position\n");
 return first;
}
 Printf("Item deleted at given Position is %d\n",
                                    cur->info);
 Prev->link = cur->link;
  freenode(cur);
  return first;
}

 void display(NODE first)
 {
  NODE temp;
   if (first==NULL)
   Printf("List is empty cannot display items\n");
   for(temp=first; temp!=NULL; temp=temp->link)
   {
    Printf("%d\n", temp->info);
   }
 }

 int main()
 {
  int item, choice, pos;
  NODE first=NULL;
```

```
for(;;)
{
 Printf("\n1: Insert_Rear \n2: Delete_Rear\n");
 Printf("3:Insert_info_Position\n4:Delete_info_
        Position\n5: Display_list\n6: Exit\n");
 Printf("Enter the choice :");
 Scanf("%d", &choice);
  switch(choice)
 {
   case 1: Printf("Enter the item at rear-end\n");
    scanf("%d", &item);
    first = insert_rear(first, item);
    break;
   case 2: first = delete_rear(first);
    break;
   case 3: Printf("Enter the item to be inserted
                   at given Position\n");
    scanf("%d", &item);
    Printf("Enter the Position:\n");
    scanf("%d", &Pos);
    first = insert_Pos(item, Pos, first);
    break;
```

```
case 4 : Printf("Enter the Position \n");
         scanf("%d",&Pos);
         first = delete_Pos(Pos, first);
         break;

         case 5 : display(first);
         break;

         default : exit(0);
         break;
    }
}

return 0;
}
```