lab program 5

```
(5) # include <stdio.h>
# include <stdlib.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 ("Memory is full \n");
  exit (0);
}
return x;
}
Void freenode (NODE x)
{
  free (x);
}
NODE insert_front (NODE first, int item)
{
 NODE temp;
 temp = getnode ();
 temp -> ~~a~~ 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 item\n");
  return first;
}
 temp = first;
temp = temp -> link;
printf ("Item deleted at the frond 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;
 return first;
```

```
NODE delete_rear (NODE first)
{
 NODE cur, prev;
 if (first == NULL)
 {
 printf ("the list is empty, cannot Delete
           item \n");
  return first;
 }
 if (first -> link == NULL)
 {
 printf("Item Deleted is : %d", first->info);
 free (first);
  return NULL;
 }
prev=NULL;
cur = first;
while (cur->link! = NULL)
 {
   prev= cur;
  cur= cur -> link;
}
printf (" Item deleted at the rear-end
is : %d", cur->info);
free(cur);
prev -> link= NULL;
return first;
}
NODE insert-pos (int item, int pos,
NODE first)
{
```

```
temp = getnode ( ) ;
temp -> info = item ;
temp -> link = NULL ;
if (first = = NULL && pos = = 1)
 return temp ;
 if (first = = NULL)
 {
printf (" Invalid position\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 ;
```

```
void display(NODE first)
{
NODE temp;
if (first == NULL)
printf("List is EMPTY, cannot Display
        Items\n");
printf("\n * * * * * \n");
for (temp=first; temp!=NULL; temp=
     temp->link)
{
printf(" %d\n",temp->info);
}
printf(" \n * * * * * * \n");
}
void main()
{
int item, choice, pos;
NODE first =NULL;
for(;;)
{
printf("\n: Insert_front \n 2: Delete_front
\n 3: Insert_rear \n 4: Delete_rear \n 5:
Insert_pos \n 6: display_list \n 7: Exit \n");
printf("Enter the choice:");
Scanf("%d", &choice);
Switch (choice)
{
case 1: printf("Enter the item at front
               -end:")
        Scanf("%d", &item);
        first = insert_front(first, item)
        break;
case 2: first = delete_front (first)
case 3: printf("Enter the item at rear end: ");
    scanf("%d", &item);
    first = insert_rear(first, item);
    break;
case 5: printf("Enter the position: ");
        scanf("%d", &pos);
        first = insert_pos(item, pos, first);
        break;
case 4: ~~printf("Enter the~~
        first = delete_rear(first);
        break;
case 6: display(first);
        break;
default: exit(0);
        break;
    }
  }
}
```