LAB PROGRAM

```
# include <stdio.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("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)
```

Scanned with CamScanner

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

```
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);
  }
}

int length(NODE first)
 {
   NODE cur;
   int count =0 ;
   if (first ==NULL)
   return 0;
   cur =first;
  while(cur!=NULL)
  {
    count ++;
    cur= cur->link;
  }
  return count;
}

void search( int key, NODE first)
{
 NODE cur;
 if (first==NULL)
 {
 printf("List is empty\n");
```

```
return;
}
 cur = first ~~*~~;
 while (cur!=NULL)
  {
    if (key == cur->info)
       break;
       cur = cur->link;
   }
    if (~~[illegible]~~ cur == NULL)
   {
      printf("Search is unsuccessful\n");
        return;
   }
    printf(" Search successful\n");
  }

NODE asc (~~[illegible]~~ node first)
  {
     NODE prev=first;
     NODE  cur = NULL;
      int temp;
   if (first == NULL)
    {
       return 0;
    }
 else
 {
  while(prev!=NULL)
  {
     cur = prev->link;
     while (cur! = NULL) {
```

```
if (prev->info > cur->info) {
   temp=prev->info;
   prev->info= cur->info;
    cur->info=temp;
   }
   cur = cur->link ;
   prev =prev->Link;
    }
  }
   return first;
  }

 NODE des (NODE first)
 {
   NODE prev=first;
   NODE cur = NULL;
      int temp;
   if (first == NULL)
   {
     return 0;
   }
   else
 {
   while (prev != NULL)
  {
   cur =prev->link;
   while (cur != NULL)
  {
   if (prev->info < cur->info)
  {
   temp=prev->info;
   prev->info = cur->info;
cur->info = temp;
 }
   cur = cur -> Link;
   }
   prev = prev->link;
   }
   }
   return first;
}

int main()
{
 int item, choice, count, key, option;
  NODE first = NULL;
   for (;;)
 {
printf("\n1: Insert_front\n 2: Delete_rear\n 3: Display_list\n 4:
   Count items\n 5: Search items\n 6: Order_list \n 7: Exit\n");
printf(" enter the choice \n");
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_rear(first);
     break;
  case 3:
    display(first);
```

```
break;
case 4:
count =length(first);
printf("Length of items in the list is %d\n", count);
break;
case 5:
printf("enter the item to be searched\n");
scanf("%d", &key);
search(key, first);
break;
case 6:
printf("\n1.ascending ordered-list\n2.descending ordered-list\n");
scanf("%d", &option);
if (option ==1)
{
  first = asc(first);
  display(first);
}

else
{
  first = des(first);
  display(first);
}
break;
default:
exit(0);
}
}
return 0;
}
```

Scanned with CamScanner