Set A

a) Implement a list library (singlylist.h) for a singly linked list with the above six operations. Write a menu driven driver program to call the operations.

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
#include "singlylist.h"
int main() {
  int option = 0,n;
DATA_NODE *list=NULL,temp_node;

  printf("Singly Linked List Example - All Operations\n");

  while (option < 7)
{
    printf("\nOptions\n");
   // printf("1 : Create list into Linked List \n");
    printf("1 : Insert into Linked List \n");
    printf("2 : Delete from Linked List \n");
    printf("3 : Display Linked List\n");
    printf("4 : Count Linked List\n");
   // printf(" : Search Linked List\n");
    printf("Others : Exit()\n");
    printf("Enter your option:");
    scanf("%d", &option);
    switch (option)
{
      case 1:
        insert();
        break;
      case 2:
        delete();
        break;
      case 3:
        display();
        break;
      case 4:
        count();
      default:
        break;
    }
  }
  return 0;
}
```

//singlylist.h

```
struct node {
  int value,data;
  struct node *next;
};

void creatlist();
void insert();
void display();
void delete();
int count();
int search();

typedef struct node DATA_NODE;

DATA_NODE *head_node, *first_node, *temp_node = 0, *prev_node, next_node;
int data;
/*DATA_NODE * createlist(DATA_NODE *list)
{
        DATA_NODE * newnode;
        int i,n,data;
        printf("\n Enter how many elements:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                newnode=(DATA_NODE *)malloc(sizeof(struct node));
                newnode->next=NULL;
                printf("\nEnter the element:");
                scanf("%d",&newnode->data);
                if(list==NULL)
                list=temp_node=newnode;
            else
            {
                temp_node->next=newnode;
                temp_node=newnode;
            }
        }
        return list;
}*/

void insert() {
  printf("\nEnter Element for Insert Linked List : \n");
  scanf("%d", &data);

  temp_node = (DATA_NODE *) malloc(sizeof (DATA_NODE));

  temp_node->value = data;

  if (first_node == 0) {
    first_node = temp_node;
  } else {
    head_node->next = temp_node;
  }
  temp_node->next = 0;
  head_node = temp_node;
  fflush(stdin);
}

void delete() {
  int countvalue, pos, i = 0;
  countvalue = count();
  temp_node = first_node;
  printf("\nDisplay Linked List : \n");

  printf("\nEnter Position for Delete Element : \n");
  scanf("%d", &pos);

  if (pos > 0 && pos <= countvalue) {
    if (pos == 1) {
      temp_node = temp_node -> next;
      first_node = temp_node;
      printf("\nDeleted Successfully \n\n");
    } else {
      while (temp_node != 0) {
        if (i == (pos - 1)) {
          prev_node->next = temp_node->next;
          if(i == (countvalue - 1))
          {
   head_node = prev_node;
   }
          printf("\nDeleted Successfully \n\n");
          break;
        } else {
          i++;
          prev_node = temp_node;
          temp_node = temp_node -> next;
        }
      }
    }
  } else
    printf("\nInvalid Position \n\n");
}

void display() {
  int count = 0;
  temp_node = first_node;
  printf("\nDisplay Linked List : \n");
  while (temp_node != 0) {
    printf("# %d # ", temp_node->value);
    count++;
    temp_node = temp_node -> next;
  }
  printf("\nNo Of Items In Linked List : %d\n", count);
}

int count() {
  int count = 0;
  temp_node = first_node;
  while (temp_node != 0) {
    count++;
    temp_node = temp_node -> next;
  }
  printf("\nNo Of Items In Linked List : %d\n", count);
  return count;
}
/*
int search(DATA_NODE *list,int num)
{
        //NODE *temp;
        for(temp_node=list;temp_node!=NULL;temp_node=temp_node->next)
                if(temp_node->data==num)
                        return temp_node;
        return NULL;
```

Set B

a) There are lists where insertion should ensure the ordering of data elements. Since the elements are in ascending order the search can terminate once equal or greater element is found. Implement a singly linked list of ordered integers (ascending/descending) with insert, search and display operations.

```
#include<stdio.h>
#include<stdlib.h>
#define newnode (struct node *)malloc(sizeof(struct node));
struct node
{
                int data;
                struct node *next;
};

struct node* create(int n)
{
                struct node *f,*s;
                int i;
                f = newnode;
                printf("\n  Enter the data :");
                scanf("%d",&f->data);
                s = f;
                for(i=2;i<=n;i++)
```

```
            {
                        s->next = newnode;
                        s = s->next;
                        printf("\n  Enter the data :");
                        scanf("%d",&s->data);
            }
            s->next=NULL;
            return f;
}

void display(struct node* f)
{
            struct node *t;
            printf("\n Link list is : \n");
            for(t=f; t!=NULL; t=t->next)
            printf("\t %d",t->data);
}

struct node *insert(struct node *f,int p)
{
            int cnt=0,i;
            struct node *t,*s;

            for(t=f; t!=NULL; t=t->next)
            {
                        cnt++;
            }

            if(p>=cnt)
            printf("\n Invalid position");
            else
            {
                        s=newnode;
                        printf("\n  Enter the data :");
                        scanf("%d",&s->data);
                        if(p==1)
                        {
                                    s->next = f;
                                    f=s;
                        }
```

```
                else
                {
                        for(i=1,t=f;i<=p-2;i++)
                        t=t->next;
                        s->next = t->next;
                        t->next = s;
                }

        }
        return f;
}

void sort(struct node *f)
{
        struct node *p,*q;
        int temp;

        for(p=f;p!=NULL;p=p->next)
        {
                for(q=p->next;q!=NULL;q=q->next)
                {
                        if(p->data > q->data)
                        {
                                temp=p->data;
                                p->data=q->data;
                                q->data=temp;
                        }
                }
        }

}

main()
{
        int n,p,p1,ch;
        struct node *l;
        do
        {
                printf("\n\n 1.create link list \n 2. dispaly \n 3.insert into link
list \n 4.sorted order \n 0.exit ");
```

```
                        printf("\n \n Enter Your choice");
                        scanf("%d",&ch);
                        switch(ch)
                        {
                                case 1 :printf("\n Enter how manu elements you wnt
to enter :");
                                        scanf("%d",&n);
                                        l=create(n);
                                        break;
                                case 2 : display(l);
                                        break;
                                case 3 :printf("\n Enter the position to enter data");
                                        scanf("%d",&p);
                                        l=insert(l,p);
                                        break;
                                case 4:sort(l);
                                        break;
                                case 0 :exit(0);
                                default : printf("\n Invalid choice");



                        }
                }while(ch!=0);


}
```

b) There are lists where new elements are always appended at the end of the list. The list can be implemented as a circular list with the external pointer pointing to the last element of the list. Implement singly linked circular list of integers with append and display operations. The operation appends (L, n), appends to the end of the list, n integers either accepted from user or randomly generated.

```
#include<stdio.h>
#include<stdlib.h>
#define newnode (struct snode*)malloc(sizeof(struct snode))
struct snode
{
        int data;
        struct snode *next;
};
struct snode *create(int no)
{
        struct snode *f,*s;
        int i;
        f=newnode;
        printf("enter data");
        scanf("%d",&f->data);
        f->next=NULL;
        s=f;
        for(i=2;i<=no;i++)
        {
        s->next=newnode;
        s=s->next;
        scanf("%d",&s->data);
        s->next=NULL;
        }
        s->next=f;
        return f;
}
void display(struct snode *f)
{
        struct snode *s;
        s=f;
        do
        {printf("\t%d",s->data);
        s=s->next;
        }while(s!=f);
}

struct node * append(struct snode *f,int n)
{       struct snode *t,*s;
        s=newnode;
        s->data=n;
        t=f;
        int i,count=0;
        do
        {       count++;
```

```
                        t=t->next;
            } while(t->next!=f);
            t->next=s;
            t=t->next;
            t->next=f;
            return f;
}

main()
{           int no,num;
            struct snode *l1,*l2;
            printf("enter no of nodes for 1st link");
            scanf("%d",&no);
            l1=create(no);
            display(l1);
            printf("\nenter data to be append : ");
            scanf("%d",&num);
            l1=append(l1,num);
            display(l1);
```