Operations in a Singly Linked list

```
//Operations in a Singly Linked list
#include<stdio.h>
#include<stdlib.h>
void insertbeg();
void insertend();
void insertpos();
void deletebeg();
void deleteend();
void deletepos();
void display();
struct node
{
  int data;
  struct node *next;
}*start=NULL;
main()
{
  int choice;
  do
    {
      printf("\nMenu:");
      printf("\n 1 Insert at Beginning");
      printf("\n 2 Insert at End");
      printf("\n 3 Insert at a Position");
      printf("\n 4 Delete from Beginning");
      printf("\n 5 Delete from End");
      printf("\n 6 Delete from a Position");
      printf("\n 7 Display");
      printf("\n 8 Exit\n Enter your choice:");
      scanf("%d",&choice);
      switch(choice)
  {
  case 1:insertbeg();
    break;
  case 2:insertend();
    break;
  case 3:insertpos();
    break;
  case 4:deletebeg();
    break;
  case 5:deleteend();
    break;
  case 6:deletepos();
    break;
  case 7:display();
    break;
```

```
  case 8:exit(0);
  default:printf("\nInvalid choice\n");
  }
    }while(choice!=8);
}

void insertbeg()
{
  int item;
  struct node *p;
  p=(struct node *)malloc(sizeof(struct node));
  printf("\nEnter the element to be inserted: ");
  scanf("%d",&item);
  p->data=item;
  p->next=start;
  start=p;
}

void insertend()
{
  int item;
  struct node *p,*temp;
  p=(struct node *)malloc(sizeof(struct node));
  printf("\nEnter the element to be inserted: ");
  scanf("%d",&item);
  temp=start;
  while(temp->next!=NULL)
    temp=temp->next;
  temp->next=p;
  p->data=item;
  p->next=NULL;
}

void insertpos()
{
  int item,pos,i;
  struct node *p,*temp;
  p=(struct node *)malloc(sizeof(struct node));
  printf("\nEnter the position: ");
  scanf("%d",&pos);
  if(pos==1)
    insertbeg();
  else
    {
      temp=start;
      for(i=1;i<(pos-1);i++)
  {
    temp=temp->next;
```

```
    if(temp==NULL)
      {
        printf("\nCannot insert\n");
        break;
      }
  }
      if(i==(pos-1))
  {
    printf("\nEnter the element to be inserted: ");
    scanf("%d",&item);
    p->next=temp->next;
    temp->next=p;
    p->data=item;
  }
    }
}

void deletebeg()
{
  struct node *temp;
  if(start==NULL)
    printf("\nThe linked list is empty\n");
  else
    {
      temp=start;
      start=start->next;
      printf("\nThe deleted element is %d\n",temp->data);
      free(temp);
    }
}

void deleteend()
{
  struct node *temp,*loc;
  if(start==NULL)
    printf("\nThe linked list is empty\n");
  else if(start->next==NULL)
    deletebeg();
  else
    {
      temp=start;
      loc=temp->next;
      while(loc->next!=NULL)
  {
    temp=temp->next;
    loc=loc->next;
  }
      temp->next=NULL;
```

```
      printf("\nThe deleted element is %d\n",loc->data);
      free(loc);
    }
}

void deletepos()
{
  struct node *temp,*loc;
  int pos,i;
  printf("\nEnter the position\t");
  scanf("%d",&pos);
  if(start==NULL)
    printf("\nLinked list is empty\n");
  else
    {
      if(pos==1)
  deletebeg();
      else
  {
    temp=start;
    loc=temp->next;
    for(i=1;i<(pos-1);i++)
      {
        temp=temp->next;
        loc=loc->next;
        if(loc==NULL)
    {
      printf("\nCannot delete\n");
      break;
    }
      }
    if(i==(pos-1))
      {
        temp->next=loc->next;
        printf("\nthe deleted element is %d\n",loc->data);
        free(loc);
      }
  }
    }
}
void display()
{
  struct node *temp;
  if(start==NULL)
    printf("\nLinked list is empty\n");
  else
    {
      temp=start;
```

```
        printf("\nThe elements in the linked list are:\n");
        while(temp!=NULL)
  {
     printf(" %d ",temp->data);
     temp=temp->next;
  }
        printf("\n");
     }
}
```

Output

```
 Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:1

Enter the element to be inserted: 1

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:1

Enter the element to be inserted: 2

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:1
```

```
Enter the element to be inserted: 3

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:7

The elements in the linked list are:
 3  2  1

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:6

Enter the position  2

the deleted element is 2

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:5

The deleted element is 1

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
```

```
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:3

Enter the position: 1

Enter the element to be inserted: 6

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:7

The elements in the linked list are:
 6  3

Menu:
 1 Insert at Beginning
 2 Insert at End
 3 Insert at a Position
 4 Delete from Beginning
 5 Delete from End
 6 Delete from a Position
 7 Display
 8 Exit
 Enter your choice:8
```

Linked list as a Stack

```
//Linked list as a Stack
#include<stdio.h>
#include<stdlib.h>
void push();
void pop();
void display();
struct node
{
  int data;
  struct node *next;
}*top=NULL;
main()
{
  int choice;
  do
    {
      printf("\nMenu:");
      printf("\n 1 Push");
      printf("\n 2 Pop");
      printf("\n 3 Display");
      printf("\n 4 Exit\n Enter your choice:");
      scanf("%d",&choice);
      switch(choice)
  {
  case 1:push();
    break;
  case 2:pop();
    break;
  case 3:display();
    break;
  case 4:exit(0);
  default:printf("\nInvalid choice\n");
  }
    }while(choice!=4);
}

void push()
{
  int item;
  struct node *p;
  p=(struct node *)malloc(sizeof(struct node));
  printf("\nEnter the element to be inserted: ");
  scanf("%d",&item);
  p->data=item;
  p->next=top;
  top=p;
```

```
}

void pop()
{
  struct node *temp;
  if(top==NULL)
    printf("\nThe linked list is empty\n");
  else
    {
      temp=top;
      top=top->next;
      printf("\nThe deleted element is %d\n",temp->data);
      free(temp);
    }
}

void display()
{
  struct node *temp;
  if(top==NULL)
    printf("\nLinked list is empty\n");
  else
    {
      temp=top;
      printf("\nThe elements in the stack are:\n");
      while(temp!=NULL)
  {
    printf(" %d ",temp->data);
    temp=temp->next;
  }
      printf("\n");
    }
}
```

Output

```
Menu:
 1 Push
 2 Pop
 3 Display
 4 Exit
 Enter your choice:1

Enter the element to be inserted: 1

Menu:
 1 Push
 2 Pop
 3 Display
```

```
 4 Exit
 Enter your choice:1

Enter the element to be inserted: 2

Menu:
 1 Push
 2 Pop
 3 Display
 4 Exit
 Enter your choice:1

Enter the element to be inserted: 3

Menu:
 1 Push
 2 Pop
 3 Display
 4 Exit
 Enter your choice:2

The deleted element is 3

Menu:
 1 Push
 2 Pop
 3 Display
 4 Exit
 Enter your choice:3

The elements in the stack are:
 2  1

Menu:
 1 Push
 2 Pop
 3 Display
 4 Exit
 Enter your choice:2

The deleted element is 2

Menu:
 1 Push
 2 Pop
 3 Display
 4 Exit
 Enter your choice:3
```

```
The elements in the stack are:
 1

Menu:
 1 Push
 2 Pop
 3 Display
 4 Exit
 Enter your choice:4
```

queue using singly linked list

```
//Queue using Single Linked list
#include<stdio.h>
#include<stdlib.h>
void insert();
void delete();
void display();

struct node
{
  int data;
  struct node *next;
}*front=NULL,*rear=NULL;

main()
{
  int choice;
  do
    {
      printf("\nMenu \n 1 Insert \n 2 Delete \n 3 Display \n 4
         Exit\n Enter your choice ");
      scanf("%d",&choice);
      switch(choice)
  {
  case 1:
    insert();
    break;
  case 2:
    delete();
    break;
  case 3:
    display();
    break;
  case 4:
    exit(0);
  default:
    printf("\nInvalid choice\n");
  }
    }while(choice!=4);
}

void insert()
{
  int item;
  struct node *p;
  p=(struct node *)malloc(sizeof(struct node));
  printf("\nEnter the element to be inserted: ");
```

```
  scanf("%d",&item);
  p->data=item;
  p->next=NULL;
  if(front==NULL)
    {
      front=p;
      rear=p;
    }
  else
    {
      rear->next=p;
      rear=p;
    }
}

void delete()
{
  struct node *temp;
  if(front==NULL)
    printf("\nThe Queue is empty\n");
  else
    {
      temp=front;
      printf("\nThe deleted element is %d\n",temp->data);
      if(front==rear)
  {
    front=NULL;
    rear=NULL;
  }
      else
  front=front->next;
      free(temp);
    }
}

void display()
{
  struct node *temp;
  if(front==NULL)
    printf("\nQueue is empty\n");
  else
    {
      temp=front;
      printf("\nThe elements in the linked list are:\n");
      while(temp!=NULL)
  {
    printf(" %d ",temp->data);
    temp=temp->next;
```

```
  }
      printf("\n");
    }
}
```

Output

```
Menu
 1 Insert
 2 Delete
 3 Display
 4 Exit
 Enter your choice 1

Enter the element to be inserted: 1

Menu
 1 Insert
 2 Delete
 3 Display
 4 Exit
 Enter your choice 1

Enter the element to be inserted: 2

Menu
 1 Insert
 2 Delete
 3 Display
 4 Exit
 Enter your choice 3

The elements in the linked list are:
 1  2

Menu
 1 Insert
 2 Delete
 3 Display
 4 Exit
 Enter your choice 2

The deleted element is 1

Menu
 1 Insert
 2 Delete
 3 Display
 4 Exit
```

```
 Enter your choice 3

The elements in the linked list are:
 2

Menu
 1 Insert
 2 Delete
 3 Display
 4 Exit
 Enter your choice 4
```