Ex 2: Doubly Linked List

REGISTER.NO:-231801155 **NAME:-SARANYA V**

DATE:-5.3.24

Program:

```
#include <stdio.h>
#include<malloc.h>
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *newnode;
struct node *head=NULL;

void insertfront(int ele)
{
    newnode=(struct node*)malloc(sizeof(struct node));
   if(head==NULL)
   {
       newnode->data=ele;
       newnode->next=NULL;
       newnode->prev=NULL;
       head=newnode;
   }
   else
   {
       newnode->data=ele;
       newnode->next=head;
       head->prev=newnode;
       head=newnode;
   }

}

void insertend(int ele)
{
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    newnode->next=NULL;
     if(head!=NULL)
     {
         struct node *t;
         t=head;
         while(t->next!=NULL)
         {
             t=t->next;
         }
```

```
        newnode->next=NULL;

        t->next=newnode;

        newnode->prev=t;

      }

      else

      {

        newnode->prev=NULL;

         head=newnode;


      }

    }

  int listsize()

    {

       int c=0;

       struct node *t;

       t=head;

       while(t!=NULL)

       {

          c=c+1;

          t=t->next;

       }

       printf("\n The size of the list is %d:\n",c);

       return c;

    }
```

```
void insertpos(int ele,int pos)
{
   int ls=0;
   struct node *temp;
   ls=listsize();
   if(head == NULL)
   {
      printf("\nInvalid position to insert a node\n");
      return;
   }
  if(head != NULL && (pos <= 0 || pos > ls))
   {
      printf("\nInvalid position to insert a node\n");
      return;
   }

 newnode=(struct node*)malloc(sizeof(struct node));

  if(newnode != NULL)
   {
      newnode->data=ele;
      temp = head;
      int count = 1;
      while(count < pos-1)
```

```
        {
           temp = temp -> next;
           count += 1;
                  }
        if(pos == 1)
        {
           newnode->next = head;
           head->prev=newnode;
           head = newnode;
        }
        else
        {
           newnode->next = temp->next;
           temp->next->prev = newnode;
           temp->next=newnode;
           newnode->prev=temp;
        }
    }
}

void find(int s)
{
    int c=1;
    struct node *temp;
    temp=head;

    if(head==NULL)

    {

       printf("\n List is empty");

    }

    else

    {


          while(temp->data!=s && temp->next!=NULL)

          {

             temp=temp->next;

             c++;

          }

          if(temp!=NULL && temp->data==s)

          {

     printf("\n Searching ele %d is present in the addr of %p in the pos%d",temp-
>data,temp,c);

    }

    else

    {

       printf("\n Searching elem %d is not present",s);

    }


}

}
```

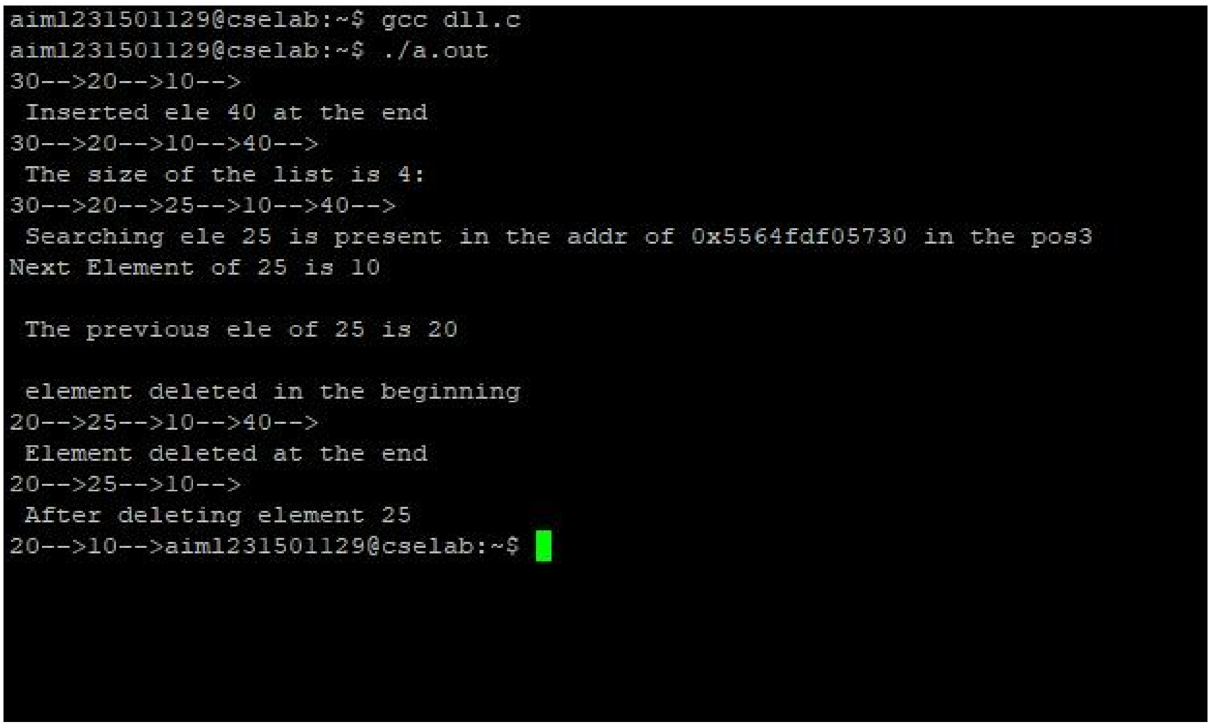
```
void findnext(int s)

{

   struct node *temp;

   temp=head;

   if(temp==NULL&&temp->next==NULL)

   {

      printf("No next element ");

   }

   else

   {

      while(temp->data!=s)

      {

         temp=temp->next;


      }

            printf("\nNext Element of %d is %d\n",s,temp->next->data);

   }


}


void findprev(int s)

{

   struct node *temp;
```

```
    temp=head;
    if(temp==NULL)
    {
        printf("List is empty ");
    }
    else
    {
        while(temp->data!=s)
        {
            temp=temp->next;

        }
          printf("\n The previous ele of %d is %d\n",s,temp->prev->data);
    }

}
void deleteAtBeginning()
{
    struct node *t;
     // t=head;
      head->next->prev=NULL;
      head=head->next;
}
void deleteAtEnd()
```

```
{
   struct node *temp;
   temp=head;
   if(head==NULL)
   {
      printf("\n List is empty");
   }
   else
   {
         while(temp->next!=NULL)
         {
            temp=temp->next;
         }
        temp->prev->next=NULL;
   }

}

void delete(int ele)
{
   struct node *t;
   t=head;
   if(t->data==ele)
   {
      head=t->next;
   }
   else
   {
   while(t->data!=ele)
   {
      t=t->next;
   }

   t->prev->next=t->next;
   t->next->prev=t->prev;

}
}



void display()
{
   struct node *temp;
   temp=head;
   while(temp!=NULL)
   {
      printf("%d-->",temp->data);
      temp=temp->next;
    }
}



int main()
{
    insertfront(10);
    insertfront(20);
    insertfront(30);
    display();
    printf("\n Inserted ele 40 at the end\n");
insertend(40);
display();
insertpos(25,3);
display();
find(25);
findnext(25);
findprev(25);
printf("\n element deleted in the beginning\n");
deleteAtBeginning();
display();
deleteAtEnd();
printf("\n Element deleted at the end\n");
display();
```

```
printf("\n After deleting element 25\n");
delete(25);
display();
    return 0;
}
```

OUTPUT:



```
aiml231501129@cselab:~$ gcc dll.c
aiml231501129@cselab:~$ ./a.out
30-->20-->10-->
 Inserted ele 40 at the end
30-->20-->10-->40-->
 The size of the list is 4:
30-->20-->25-->10-->40-->
 Searching ele 25 is present in the addr of 0x5564fdf05730 in the pos3
Next Element of 25 is 10

 The previous ele of 25 is 20

 element deleted in the beginning
20-->25-->10-->40-->
 Element deleted at the end
20-->25-->10-->
 After deleting element 25
20-->10-->aiml231501129@cselab:~$
```