**Program :**

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node \*next;

    struct node \*prev;

}\*head=NULL,\*newnode,\*temp,\*temp1;

struct node \*temp2,\*new,\*temp3,\*p,\*flag,\*flag1,\*i,\*j;

void create()

{

    int n;

    printf("Enter number of nodes : ");

    scanf("%d",&n);

    while(n>0)

    {

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

        printf("Enter data of node : ");

        scanf("%d",&newnode->data);

        newnode->next=NULL;

        newnode->prev=NULL;

        if(head==NULL)

        {

            head=newnode;

        }

        else

        {

            temp=head;

            while(temp->next!=NULL)

            {

                temp=temp->next;

            }

            temp->next=newnode;

            newnode->prev=temp;

        }

        n--;

    }

}

void insertatfront()

{

    temp1=(struct node\*) malloc(sizeof(struct node));

    printf("\nEnter data : ");

    scanf("%d", &temp1->data);

    temp1->next = NULL;

    temp1->prev=NULL;

    if(head==NULL)

    {

    head=temp1;

}

else

{

temp1->next=head;

head->prev=temp1;

head=temp1; }

    printf("Updated list : ");

    temp=head;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->next;

    }

}

void insertatend()

{

     temp2=(struct node\*) malloc(sizeof(struct node));

     printf("\nEnter data : ");

     scanf("%d", &temp2->data);

     temp2->next = NULL;

     temp2->prev=NULL;

     if(head==NULL)

     {

      head=temp2;

}

else

{

     temp=head;

     while (temp->next != NULL)

     {

     temp=temp->next;

     }

     temp->next = temp2;

     temp2->prev=temp;

}

     printf("Updated list : ");

     temp=head;

     while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->next;

    }

}

void insertatposition()

{

    int pos,i=1;

    new =(struct node\*)malloc(sizeof(struct node));

    printf("\nEnter position and data :");

    scanf("%d %d", &pos, &new->data);

    new->next = NULL;

    new->prev=NULL;

    if(head==NULL)

    {

    head=new;

}

else

{

temp3=head;

    while (i < pos - 1)

    {

    temp3=temp3->next;

i++;

    }

    new->next=temp3->next;

    new->prev=temp3;

    temp3->next->prev=new;

    temp3->next = new;

    }

    printf("Updated list : ");

    temp=head;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->next;

    }

}

void deleteatpos()

{

    struct node \*flag;

    int pos,i=1;

    if(head==NULL)

    {

    printf("List is Empty.\n");

}

else

{

    printf("Enter position of linked list to delete : ");

    scanf("%d",&pos);

    temp=head;

    while(i<pos)

    {

      flag=temp;

      temp=temp->next;

      i++;

    }

    temp->prev->next=temp->next;

    temp->next->prev=temp->prev;

    free(temp);

    printf("Updated list : ");

    temp=head;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->next;

    }

}

}

void deleteatend()

{

if(head==NULL)

{

printf("List is Empty.\n");

}

else

{

    flag1=head;

    while(flag1->next!=NULL)

    {

        p=flag1;

        flag1=flag1->next;

    }

    flag1->prev->next=NULL;

    free(flag1);

}

    printf("Updated list : ");

    temp=head;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->next;

    }

}

void deleteatfront()

{

if(head==NULL)

{

printf("List is empty.\n");

}

else

{

    temp=head;

    head=head->next;

    head->prev=NULL;

    free(temp);

}

    printf("Updated list : ");

    temp=head;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->next;

    }

}

void display()

{

    printf("Linked list : ");

    temp=head;

    while(temp!=NULL)

    {

        printf("%d ",temp->data);

        temp=temp->next;

    }

}

int search()

{

    int pos=0,key;

    printf("Enter the element to search: ");

    scanf("%d", &key);

    flag=head;

    while(flag!=NULL)

    {

    if(flag->data==key)

      {

        return pos;

      }

    flag=flag->next;

    pos++;

     }

   return -1;

}

void sort()

{

      int num;

      i=head;

      while(i!=NULL)

       {

          j=i->next;

          while(j!=NULL)

             {

              if(i->data>j->data)

              {

                  num=i->data;

                  i->data=j->data;

                  j->data=num;

              }

                j=j->next;

             }

          i=i->next;

       }

    printf("Linked list sorted\n");

}

int main()

{

    int choice,choice1,choice2,ans;

    do

    {

        printf("\nEnter (1) to create linked list (2) to insert a element (3) to delete a element (4) to search (5) to sort (6) to display (7) to exit.\n");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

               create();

               break;

            case 2:

               do

               {

                printf("\nEnter (1) to insert element at front (2) to insert element at end (3) to insert element at specified position (4) to exit.\n");

               scanf("%d",&choice1);

                switch(choice1)

                {

                    case 1:

                     insertatfront();

                     break;

                    case 2:

                     insertatend();

                     break;

                    case 3:

                     insertatposition();

                     break;

                    case 4:

                     printf("EXIT");

                     break;

                    default:

                     printf("Error");

                     break;

                }

               } while(choice1!=4);

               break;

            case 3:

                do

               {

                printf("\nEnter (1) to delete element at front (2) to delete element at end (3) to delete element at specified position (4) to exit.\n");

               scanf("%d",&choice2);

                switch(choice2)

                {

                    case 1:

                     deleteatfront();

                     break;

                    case 2:

                     deleteatend();

                     break;

                    case 3:

                     deleteatpos();

                     break;

                    case 4:

                     printf("EXIT");

                     break;

                    default:

                     printf("Error");

                     break;

                }

               } while(choice2!=4);

              break;

           case 4:

              ans=search();

              if(ans==1)

                {

              printf("Element not founnd");

                }

               else

                 {

              printf("Element found at position %d", ans+1);

                 }

              break;

           case 5:

              sort();

              break;

            case 6:

             display();

             break;

            case 7:

             printf("EXIT");

             break;

            default:

             printf("ERROR");

             break;

        }

    }

    while(choice!=7);

    return 0;

}

**Output :**

PS C:\Users\hp\OneDrive\Desktop\coding> cd "c:\Users\hp\OneDrive\Desktop\coding\DS-2\" ; if ($?) { gcc doubly\_linked.c -o doubly\_linked } ; if ($?) { .\doubly\_linked }

Enter (1) to create linked list (2) to insert a element (3) to delete a element (4) to search (5) to sort (6) to display (7) to exit.

1

Enter number of nodes : 3

Enter data of node : 98

Enter data of node : 45

Enter data of node : 32

Enter (1) to create linked list (2) to insert a element (3) to delete a element (4) to search (5) to sort (6) to display (7) to exit.

2

Enter (1) to insert element at front (2) to insert element at end (3) to insert element at specified position (4) to exit.

1

Enter data : 99

Updated list : 99 98 45 32

Enter (1) to insert element at front (2) to insert element at end (3) to insert element at specified position (4) to exit.

4

EXIT

Enter (1) to create linked list (2) to insert a element (3) to delete a element (4) to search (5) to sort (6) to display (7) to exit.

5

Linked list sorted

Enter (1) to create linked list (2) to insert a element (3) to delete a element (4) to search (5) to sort (6) to display (7) to exit.

6

Linked list : 32 45 98 99

Enter (1) to create linked list (2) to insert a element (3) to delete a element (4) to search (5) to sort (6) to display (7) to exit.

7

EXIT

PS C:\Users\hp\OneDrive\Desktop\coding\DS-2>