```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   struct node *prev;
   int data;
   struct node *next;
   }*head,*temp,*temp1,*temp2;

void insert_beg();
void insert_end();
void insert_mid();
void delete();
void delete_forward();
void delete_backward(int i);

int count=0;

 void main()
  {
  int choice,insert_option,print_
option;
  printf("\n\n Welcome to the Implementation of Doubly linked list\n");
  do
  {
  printf("\n Please select an operation to perform from the below list");
  printf("1.Insert a node \n 2.delete a node \n 3.print the existing list\n 4.exit \n");
  printf("Enter your choice: ");
  scanf("%d",&choice);
  printf("\n \n");
switch(choice)
  {
  case 1:
  do
  {
    printf("select a position where you want to insert a new node\n");
printf("1.Beginning of the list\n 2.at the end of the list\n 3.Insert in between\n 4.Exit the insert
option\n");
  printf("Enter the choice:");
scanf("%d",&insert_option);
switch(insert_option){
case 1:  insert_beg();
break;
```

```
case 2:  insert_end();
break;
case 3:  insert_mid();
break;
case 4:  printf("insert operation exit");
break;
default: printf("Please enter a valid choice");
break;
}
}while(insert_option !=4);
printf("\n\n");
break;

case 2:delete();
break;
case 3:do
{
printf("--Display option menu--\n");
printf("1. Print list forward direction\t 2.Print list in backward direction \t 3.Exit\n");
printf("Enter an option:");
scanf("%d",&print_option);
switch(print_option)
{
case 1: display_forward();
printf("\n\n");
break;
case 2:temp2=head;
if(temp2==NULL)
{
printf("Error: list is empty to display\n");
}
else{
printf("Linked list elements in backward direction\n");
display_backward(temp2->data);
}
printf("\n\n");
break;
case 3:printf("Print operation exit!\n");
break;
default:printf("Please enter a valid choice : 1,2,3\n");
break;
}
}while(print_option!=3);
printf("\n\n");
```

```
break;
case 4:printf("Exit program finished!;");
break;
default:printf("Please enter a valid choice : 1.,2.,3.,4");
break;
}
}while(choice!=4);
}



void create()
{
int x;
temp=(struct node *)malloc(1*sizeof(struct node));
temp->prev=NULL;
temp->next=NULL;
printf("Enter the data to be inserted:");
scanf("%d",&x);
printf("\n");
temp->data=x;
count++;
}

void insert_beg(){
if(head==NULL){
create();
head=temp;
temp1=head;
}
else{
create();
temp->next=head;
head->prev=temp;
head=temp;
}
}


void insert_end(){
if(head==NULL){
create();
head=temp;
temp1=head;
```

```
}
else{
create();
temp1->next=temp;
temp->prev=temp1;
temp1=temp;
}
}

void insert_mid()
{
  int pos,i=2;
  printf("ENter position of the element to be inserted: ");
  scanf("%d",&pos);
  temp2 = head;

  if((pos < 1) || (pos >= count + 1))
  {
    printf("\nPosition out of range to insert");
    return;
  }
  if((head == NULL) && (pos != 1))
  {
    printf("\nEmpty list cannot insert other than 1st position");
    return;
  }
  if((head == NULL) && (pos == 1))
  {
    create();
    head = temp;
    temp1 = head;
    return;
  }
  else
  {
    while(i < pos)
    {
      temp2 = temp2 -> next;
      i++;
    }
    create();
    temp -> prev = temp2;
    temp -> next = temp2 -> next;
    temp2 -> next -> prev = temp;
```

```
    temp2 -> next = temp;
  }
}

void delete()
{
  int pos,i=1;
  printf("Enter position of the element to be declared: ");
  scanf("%d",&pos);
  temp2 = head;

  if((pos<1) || (pos>=count+1))
  {
    printf("Error: Position out of range to delete \n");
    return;
  }
  if(head == NULL)
  {
    printf("Error: Empty list no elements to delete \n");
    return;
  }
  else
  {
    while(i<pos)
    {
      temp2=temp2 -> next;
      i++;
    }
    if(i == 1)
    {
      if(temp2 -> next == NULL)
      {
        printf("Node deleted from list\n");
        free(temp2);
        temp2=head=NULL;
        return;
      }
    }
    if(temp2 -> next == NULL)
    {
      temp2 -> prev -> next=NULL;
      free(temp2);
      printf("Node deleted from list");
      return;
```

```
    }
    temp2 -> next -> prev = temp2 -> prev;
    if(i != 1)
    {
      temp2 -> prev -> next = temp2 -> next;
    }
    if(i==1)
    {
      head = temp2 -> next;
    }
    printf("Node deleted from list\n");
    free(temp2);
  }
  count--;
}

void display_forward()
{
  temp2=head;
  if(temp==NULL)
  {
    printf("List empty to display\n");
    return;
  }
  printf("Linked list elements in forward direction: ");

  while(temp2 -> next != NULL)
  {
    printf("%d",temp2 -> data);
    temp2=temp2 -> next;
  }
  printf("%d",temp2 -> data);
}

void display_backward(int i)
{
  if(temp2 != NULL)
  {
    i = temp2 -> data;
    temp2 = temp2 -> next;
    display_backward(i);
    printf("%d",i);
  }
}
```

Output:

```
dl406@itadmin:~/Desktop$ ./a.out


 Welcome to the Implementation of Doubly linked list

 Please select an operation to perform from the below list1.Insert a node
 2.delete a node
 3.print the existing list
 4.exit
Enter your choice: 1


select a position where you want to insert a new node
1.Beginning of the list
 2.at the end of the list
 3.Insert in between
 4.Exit the insert option
Enter the choice:1
Enter the data to be inserted:13

select a position where you want to insert a new node
1.Beginning of the list
 2.at the end of the list
 3.Insert in between
 4.Exit the insert option
Enter the choice:2
Enter the data to be inserted:16

select a position where you want to insert a new node
1.Beginning of the list
 2.at the end of the list
 3.Insert in between
 4.Exit the insert option
Enter the choice:1
Enter the data to be inserted:1

select a position where you want to insert a new node
1.Beginning of the list
 2.at the end of the list
 3.Insert in between
 4.Exit the insert option
Enter the choice:4
insert operation exit
```

```
select a position where you want to insert a new node
1.Beginning of the list
 2.at the end of the list
 3.Insert in between
 4.Exit the insert option
Enter the choice:4
insert operation exit


 Please select an operation to perform from the below list1.Insert a node
 2.delete a node
 3.print the existing list
 4.exit
Enter your choice: 3


--Display option menu--
1. Print list forward direction  2.Print list in backward direction      3.Exit
Enter an option:1
Linked list elements in forward direction: 11316

--Display option menu--
1. Print list forward direction  2.Print list in backward direction      3.Exit
Enter an option:3
Print operation exit!



 Please select an operation to perform from the below list1.Insert a node
 2.delete a node
 3.print the existing list
 4.exit
Enter your choice: 2


Enter position of the element to be declared: 2
Node deleted from list

 Please select an operation to perform from the below list1.Insert a node
 2.delete a node
 3.print the existing list
 4.exit
Enter your choice: 3
```

```
--Display option menu--
1. Print list forward direction  2.Print list in backward direction      3.Exit
Enter an option:1
Linked list elements in forward direction: 116

--Display option menu--
1. Print list forward direction  2.Print list in backward direction      3.Exit
Enter an option:3
Print operation exit!



 Please select an operation to perform from the below list1.Insert a node
 2.delete a node
 3.print the existing list
 4.exit
Enter your choice: 4


Exit program finished!;dl406@itadmin:~/Desktop$
```