```c
/*****************************************************************************
 C program for the implementation and performing operations on
                    DOUBLE LINKED LIST
*****************************************************************************/
#include<stdio.h>
#include<stdlib.h>
int create();
int add_beg();
int add_end();
int add_after();
int add_before();
int add_pos();
int delete();
int display_forward();
int display_reverse();
int search();
/*******************************************************************
Structure of a node in Double Linked list (DLL).
*******************************************************************/
struct node
	{
	struct node *prev;
	int info;
	struct node *next;
	}*new;

struct node *start=NULL;
struct node *end=NULL;

int data,item,pos;

int main()
{
	int ch;
	while(1)
	{
	printf("\n\n****Doubly Linked List****");
	printf("\n1.Create");
	printf("\n2.Insert Node at Beginning");
	printf("\n3.Insert Node at End");
	printf("\n4.Add After Node");
	printf("\n5.Add Before Node");
	printf("\n6.Add at Position");
	printf("\n7.Delete Node");
	printf("\n8.Display List in Forward direction");
	printf("\n9.Display List in Reverse direction");
	printf("\n10.Search Element");
```

```
        printf("\n\tEnter Your Choice : ");
        scanf("%d",&ch);


                switch(ch)
                {
                        case 1: create();
                                        break;
                        case 2: add_beg();
                                        break;
                        case 3: add_end();
                                        break;
                        case 4: add_after();
                                        break;
                        case 5: add_before();
                                        break;
                        case 6: add_pos();
                                        break;
                        case 7: delete();
                                        break;
                        case 8: display_forward();
                                        break;
                        case 9: display_reverse();
                                        break;
                        case 10: search();
                                        break;
                        default:exit(1);
                }
        }
        return 0;
}

/*******************************************************************
Create DLL
*******************************************************************/
int create()
{
        int i,n;
        if(start!=NULL)
        {
                printf("Link is already present, you can not create again");
                return 0;
        }

        add_beg();
}
```

```
/*****************************************************************
Add a node in the beginning
*****************************************************************/
int add_beg()
{
        printf("\nEnter the Element to be inserted: ");
        scanf("%d",&data);
        new=(struct node *)malloc(sizeof(struct node));
        new->info=data;
        if(end==NULL)
        {
                new->next=start;
                new->prev=end;
                start=new;
                end=new;
                return 0;
        }
        else
        {
                new->next=start;
                new->prev=NULL;
                start->prev=new;
                start=new;
                return 0;
        }

}
/*****************************************************************
Add a node at end
*****************************************************************/

int add_end()
{
        printf("\nEnter the Element to be inserted: ");
        scanf("%d",&data);
        new=(struct node *)malloc(sizeof(struct node));
        new->info=data;
        new->next=NULL;
        new->prev=end;
        end->next=new;
        end=new;
        return 0;
}
```

```
/*****************************************************************
Add a node after a particular node
*****************************************************************/
int add_after()
{
	printf("\nEnter the Element to be inserted: ");
	scanf("%d",&data);
	printf("Enter the Element after which to insert : ");
	scanf("%d",&item);
	struct node *temp;
	temp=start;
	while(temp!=NULL)
	{
		while(temp->info!=item)
		{
			temp=temp->next;
		}
		new=(struct node *)malloc(sizeof(struct node));
		new->info=data;
			new->prev=temp;
			new->next=temp->next;
			temp->next->prev=new;
			temp->next=new;
		return 0;
	}
	printf("\n\t%d is not present in the list",item);
	return 0;
}
```

```
/*********************************************************************
Add anode before particular node
*********************************************************************/
int add_before()
{
	if(start==NULL)
		{
			printf("\nList is empty");
			return 0;
		}
	if(item==start->info)
		{
			add_beg();
		}
	printf("\nEnter the Element to be inserted: ");
	scanf("%d",&data);
	printf("\nEnter the Element before which to insert : ");
	scanf("%d",&item);
	struct node *new,*temp;
	temp=start;
	while(temp->next!=NULL)
	{
		while(temp->next->info!=item)
		{
			temp=temp->next;
		}
		new=(struct node *)malloc(sizeof(struct node));
		new->info=data;
		new->prev=temp;
		new->next=temp->next;
		temp->next->prev=new;
		temp->next=new;
		return 0;
	}
	printf("\n\t%d item is not present in the list",item);
	return 0;
}
```

```
/*******************************************************************
Add a node at a particular position
*******************************************************************/

int add_pos()
{
	struct node *new,*temp;
	int i;
	temp=start;
	printf("\nEnter the Element to be inserted: ");
	scanf("%d",&data);
	printf("\nEnter the position at which to insert: ");
	scanf("%d",&pos);
	for(i=1;i<pos-1 && pos!=0;i++)
		temp=temp->next;
	if(temp==NULL)
		printf("\n\tThere are less than %d elements",pos);
	else
	{
		new=(struct node *)malloc(sizeof(struct node));
		new->info=data;
		if(pos==1)
		{
			start->prev=new;
			new->next=start;
			start = new;
		}
		else
		{
			new->prev=temp;
			new->next=temp->next;
			temp->next->prev=new;
			temp->next=new;
		}
	}
	return 0;
}
```

```
/*********************************************************************
Delete a node
*********************************************************************/
int delete()
{
	struct node *new,*temp;
	printf("\nEnter the element to be deleted : ");
	scanf("%d",&item);
	if(start==NULL)
	{
		printf("\n\tList is Empty");
		return 0;
	}

	else if(start->info==item)
	{
		new=start;
		start=start->next;
		start->prev=NULL;
		free(new);
		return 0;
	}
	else if(end->info==item)
	{
		new=end;
		end=end->prev;
		end->next=NULL;
		free(new);
		return 0;
	}
	temp=start;
	while(temp->next!=NULL)
	{
		if(temp->next->info==item)
		{
			new=temp->next;
			temp->next=new->next;
			new->next->prev=temp;
			free(new);
			return 0;
		}
	temp=temp->next;
	}
	printf("\n\t%d item is not present in the list",item);
	return 0;
}
```

```
/*****************************************************************
Display/traverse list from start to end
*****************************************************************/
int display_forward()
{
	struct node *temp;
	if(start==NULL || end==NULL)
	{
		printf("\n\tList is Empty");
		return 0;
	}
	temp=start;
	printf("\nForward List : \n");
	while(temp!=NULL)
	{
	printf("Prev Addr=%d\tData=%d\tNext Addr=%d\n",temp->prev,temp->info,temp->next);
		temp=temp->next;
	}
	printf("\n\n");
}

/*****************************************************************
Display/traverse a list from end to start
*****************************************************************/

int display_reverse()
{
	struct node *temp;
	if(start==NULL || end==NULL)
	{
		printf("\n\tReverse List is Empty");
		return 0;
	}
	temp=end;
	printf("\nList : ");
	while(temp!=NULL)
	{
	printf("Prev Addr=%d\tData=%d\tNext Addr=%d\n",temp->prev,temp->info,temp->next);
		temp=temp->prev;
	}
	printf("\n\n");
}
```

```
/***************************************************************
Search for a node
***************************************************************/

int search()
{
	struct node *temp=start;
	int pos=1;
	printf("\nEnter the Element to search: ");
	scanf("%d",&item);
	while(temp!=NULL)
	{
		if(temp->info==item)
		{
			printf("\n\tItem %d is found at position %d",item,pos);
			return 0;
		}
		temp=temp->next;
		pos++;
	}
	printf("\n\ttem %d is not found in the list",item);
}
```