Qn. Insertion and deletion in a doubly linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	struct node *prev;
	struct node *next;
	int data;
};
struct node *head;
void insertbeg();
void insertlast();
void insertspecific();
void deletebeg();
void deletelast();
void deletespecific();
void display();
void search();
void main ()
{
int choice =0;
	while(choice != 9)
	{
		printf("\n choose option");
		printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n5.Delete from last\n6.Delete the node after the given data\n7.Search\n8.Display\n9.Exit\n");
		printf("\nEnter your choice?\n");
		scanf("\n%d",&choice);
		switch(choice)
		{
			case 1:
				insertbeg();
				break;
			case 2:
				insertlast();
				break;
			  case 3:
				insertspecific();
				break;
			case 4:
				deletebeg();
				break;
			case 5:
				deletelast();
				break;
```

```
                case 6:
                        deletespecific();
                        break;
                case 7:
                        search();
                        break;
                case 8:
                        display();
                        break;
                case 9:
                        exit(0);
                        break;
                default:
                        printf("Enter valid choice..");
            }
       }
}
void insertbeg()
{
     struct node *ptr;
     int item;
     ptr = (struct node *)malloc(sizeof(struct node));
     if(ptr == NULL)
     {
           printf("\nOverflow");
     }
     else
     {
       printf("\nEnter item : ");
       scanf("%d",&item);

     if(head==NULL)
     {
           ptr->next = NULL;
           ptr->prev=NULL;
           ptr->data=item;
           head=ptr;
     }
     else
     {
           ptr->data=item;
           ptr->prev=NULL;
           ptr->next = head;
           head->prev=ptr;
           head=ptr;
     }
     printf("\nNode inserted : \n");
```

```
}

}
void insertlast()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node *) malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOverflow");
    }
    else
    {
        printf("\nEnter value");
        scanf("%d",&item);
          ptr->data=item;
        if(head == NULL)
        {
            ptr->next = NULL;
            ptr->prev = NULL;
            head = ptr;
        }
        else
        {
            temp = head;
            while(temp->next!=NULL)
            {
                temp = temp->next;
            }
            temp->next = ptr;
            ptr ->prev=temp;
            ptr->next = NULL;
            }

        }
      printf("\nInsertion Successful\n");
     }
void insertspecific()
{
    struct node *ptr,*temp;
    int item,loc,i;
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\n Overflow");
    }
```

```
    else
    {
        temp=head;
        printf("Enter location");
        scanf("%d",&loc);
        for(i=0;i<loc;i++)
        {
            temp = temp->next;
            if(temp == NULL)
            {
                printf("\n There are less than %d elements", loc);
                return;
            }
        }
        printf("Enter value");
        scanf("%d",&item);
        ptr->data = item;
        ptr->next = temp->next;
        ptr -> prev = temp;
        temp->next = ptr;
        temp->next->prev=ptr;
        printf("\nInsertion Successful\n");
    }
}
void deletebeg()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n Underflow");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nDeletion Successful\n");
    }
    else
    {
        ptr = head;
        head = head -> next;
        head -> prev = NULL;
        free(ptr);
        printf("\nDeletion Successful\n");
    }

}
```

```
void deletelast()
{
	struct node *ptr;
	if(head == NULL)
	{
		printf("\n Underflow");
	}
	else if(head->next == NULL)
	{
		head = NULL;
		free(head);
		printf("\nDeletion Successful\n");
	}
	else
	{
		ptr = head;
		if(ptr->next != NULL)
		{
			ptr = ptr -> next;
		}
		ptr -> prev -> next = NULL;
		free(ptr);
		printf("\nDeletion Successful\n");
	}
}
void deletespecific()
{
	struct node *ptr, *temp;
	int val;
	printf("\n Enter the data after which the node is to be deleted : ");
	scanf("%d", &val);
	ptr = head;
	while(ptr -> data != val)
	ptr = ptr -> next;
	if(ptr -> next == NULL)
	{
		printf("\nCan't delete\n");
	}
	else if(ptr -> next -> next == NULL)
	{
		ptr ->next = NULL;
	}
	else
	{
		temp = ptr -> next;
		ptr -> next = temp -> next;
		temp -> next -> prev = ptr;
```

```
            free(temp);
            printf("\nDeletion Successful\n");
        }
}
void display()
{
        struct node *ptr;
        printf("\n The elements are : \n");
        ptr = head;
        while(ptr != NULL)
        {
            printf("%d\n",ptr->data);
            ptr=ptr->next;
        }
}
void search()
{
        struct node *ptr;
        int item,i=0,flag;
        ptr = head;
        if(ptr == NULL)
        {
            printf("\nEmpty List\n");
        }
        else
        {
            printf("\nEnter item which you want to search?\n");
            scanf("%d",&item);
            while (ptr!=NULL)
            {
                if(ptr->data == item)
                {
                    printf("\nItem found at location %d ",i+1);
                    flag=0;
                    break;
                }
                else
                {
                    flag=1;
                }
                i++;
                ptr = ptr -> next;
            }
            if(flag==1)
            {
                printf("\nItem not found\n");
            }
```

```
        }

}
```

OUTPUT

```
choose option
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Display
9.Exit

Enter your choice?
1

Enter item : 4

Node inserted :

 choose option
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Display
9.Exit

choose option
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Display
9.Exit

Enter your choice?
1

Enter item : 4

Node inserted :
```

choose option
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Display
9.Exit

Enter your choice?
2

Enter value6

Insertion Successful

choose option
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Display
9.Exit

Enter your choice?
8

The elements are :
4
6

choose option
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Display
9.Exit

Enter your choice?
9