

1)INSERTION,DELETION,DISPLAY,SEARCH OPERATION IN DOUBLY LINKED LIST

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

struct node
{
    struct node *prev;
    struct node *next;
    int data;
};

struct node *head;

void insertion_beginning();
void insertion_last();
void insertion_specified();
void deletion_beginning();
void deletion_last();
void deletion_specified();
void display();
void search();

void main ()
{
    int choice =1;
    while(choice!= 9)
    {
        printf("\n*****Main Menu*****\n");
        printf("\nChoose one option from the following list ...\n");
```

```

printf("\n=====\\n");

printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from
Beginning\n 5.Delete from last\n6.Delete the node after the given data\n7.Search\n8.Show\n9.Exit\\n");

printf("\nEnter your choice?\\n");

scanf("\\n%d",&choice);

switch(choice)
{
    case 1:
        insertion_beginning();
        break;
    case 2:
        insertion_last();
        break;
    case 3:
        insertion_specified();
        break;
    case 4:
        deletion_beginning();
        break;
    case 5:
        deletion_last();
        break;
    case 6:
        deletion_specified();
        break;
    case 7:
        search();
        break;
    case 8:

```

```

        display();

        break;

    case 9:

        exit(0);

        break;

    default:

        printf("Please enter valid choice..");

    }

}

void insertion_beginning()

{

    struct node *ptr;

    int item;

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

    if(ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

    else

    {

        printf("\nEnter Item value");

        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 insertion_last()
{
    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("\nnode inserted\n");
}
void insertion_specified()
{
    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 the 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("\nnode inserted\n");
}
}

void deletion_beginning()
{
    struct node *ptr;
    if(head == NULL)
    {

```

```

        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        head = head -> next;
        head -> prev = NULL;
        free(ptr);
        printf("\nnode deleted\n");
    }

}

void deletion_last()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
    }
}

```

```

        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nnode deleted\n");
    }
}

void deletion_specified()
{
    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("\nnode deleted\n");
    }
}

void display()
{
    struct node *ptr;
    printf("\n printing values...\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("\n Item not found\n");
        }
    }
}
```

OUTPUT

****Main Menu****

Choose one option from the following list ...

=====

- 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.Show
- 9.Exit

Enter your choice?

1

Enter Item value30

Node inserted

****Main Menu****

Choose one option from the following list ...

=====

- 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.Show
- 9.Exit

Enter your choice?

2

Enter value40

node inserted

****Main Menu****

Choose one option from the following list ...

=====

- 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.Show
- 9.Exit

Enter your choice?

3

Enter the location50

There are less than 50 elements

****Main Menu****

Choose one option from the following list ...

=====

- 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.Show
- 9.Exit

Enter your choice?

4

node deleted

****Main Menu****

Choose one option from the following list ...

=====

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.Show

9.Exit

Enter your choice?

7

Enter item which you want to search?

50

Item not found

****Main Menu****

Choose one option from the following list ...

=====

- 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.Show
- 9.Exit

Enter your choice?

7

Enter item which you want to search?

40

item found at location 1

****Main Menu****

Choose one option from the following list ...

=====

- 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.Show
- 9.Exit

Enter your choice?

8

printing values...

40

****Main Menu****

Choose one option from the following list ...

=====

- 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.Show

9.Exit

Enter your choice?

