Experiment 4

Objective: To implement Linked list in C

Code:

```
 #include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node*next;
};
struct node*head;
void insert_beg();
void insert_last();
void insert_random();
void delete_beg();
void delete_last();
void delete_random();
void search();
void display();
void main()
{
   int n=0;
   while(n!=9)
   {
      printf("\n************MENU*************");
      printf("\n1.Insert in begining");
      printf("\n2.Insert at last");
```

```
printf("\n3.Inseert at random location");
printf("\n4.Delete from begining");
printf("\n5.Delete from last");
printf("\n6.Delete at random location");
printf("\n7.Search for element");
printf("\n8.Display");
printf("\n9.Exit");
printf("\nEnter your choice:");
scanf("%d",&n);
switch(n)
{
   case 1:
   insert_beg();
   break;
   case 2:
   insert_last();
   break;
   case 3:
   insert_random();
   break;
   case 4:
   delete_beg();
   break;
   case 5:
   delete_last();
   break;
   case 6:
```

```
            delete_random();
            break;
            case 7:
            search();
            break;
            case 8:
            display();
            break;
            case 9:
            exit(0);
            break;
            default:
            printf("Please enter valid choice");
        }
    }
}
void insert_beg()
{
    struct node*ptr;
    int item;
    ptr=(struct node*)malloc(sizeof(struct node*));
    if(ptr==NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
```

```
        printf("\nEnter value");
        scanf("%d",&item);
        ptr->data=item;
        ptr->next=head;
        head=ptr;
        printf("\nNode inserted");
    }
}
void insert_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;
            head=ptr;
            printf("\nNode inserted");
```

```
        }
        else
        {
            temp=head;
            while(temp->next!=NULL)
            {
                temp=temp->next;
            }
            temp->next=ptr;
            ptr->next=NULL;
            printf("\nNode inserted");
        }
    }
}
void insert_random()
{
    int i,loc,item;
    struct node*ptr,*temp;
    ptr=(struct node*)malloc(sizeof(struct node));
    if(ptr==NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter element value");
        scanf("%d",&item);
```

```
        ptr->data=item;
        printf("\nEnter the location which you want to insert");
        scanf("%d",&loc);
        temp=head;
        for(i=0;i<loc;i++)
        {
            temp=temp->next;
            if(temp==NULL)
            {
                printf("\nCan't insert");
                return;
            }
        }
        ptr->next=temp->next;
        temp->next=ptr;
        printf("\nNode inserted");
    }
}
void delete_beg()
{
    struct node*ptr;
    if(head==NULL)
    {
        printf("\nList is empty");
    }
    else
    {
```

```
        ptr=head;
        head=ptr->next;
        free(ptr);
        printf("\nNode deleted in begining");
    }
}
void delete_last()
{
    struct node*ptr,*ptr1;
    if(head==NULL)
    {
        printf("\nList is empty");
    }
    else if(head->next==NULL)
    {
        head==NULL;
        free(head);
        printf("\nOnly node of list deleted");
    }
    else
    {
        ptr=head;
        while(ptr->next!=NULL)
        {
            ptr1=ptr;
            ptr=ptr->next;
        }
```

```
        ptr1->next==NULL;
        free(ptr);
        printf("\nDeleted node from last");
    }
}
void delete_random()
{
    struct node*ptr,*ptr1;
    int loc,i;
    printf("\nEnter the location of node to delete");
    scanf("%d",&loc);
    ptr=head;
    for(i=0;i<loc;i++)
    {
        ptr1=ptr;
        ptr=ptr->next;
        if(ptr==NULL)
        {
            printf("\nCan't delete");
            return;
        }
    }
    ptr1->next=ptr->next;
    free(ptr);
    printf("\nDeleted node %d",loc+1);
}
void search()
```

```
{
    struct node*ptr;
    int item,i=0,flag;
    ptr=head;
    if(ptr==NULL)
    {
        printf("\nEmpty List");
    }
    else
    {
        printf("\nEnter item to search");
        scanf("%d",&item);
        while(ptr!=NULL)
        {
            if(ptr->data==item)
            {
                printf("Item found at %d",i+1);
                flag=0;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr=ptr->next;
        }
        if(flag==1)
```

```
        {
            printf("\nItem not found");
        }
    }
}
void display()
{
    struct node*ptr;
    ptr=head;
    if(ptr==NULL)
    {
        printf("Nothing to display");
    }
    else
    {
        printf("\nPrinting values:");
        while(ptr!=NULL)
        {
            printf("\n%d",ptr->data);
            ptr=ptr->next;
        }
    }
}
```

Output:

```
***********MENU*************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:1

Enter value1

Node inserted
***********MENU*************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:2

Enter value:4

Node inserted
***********MENU*************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
```

```
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:3

Enter element value2

Enter the location which you want to insert1

Node inserted
***********MENU*************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:8

Printing values:
1
4
2
***********MENU*************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
```

```
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:4

Node deleted in begining
***********MENU************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:8

Printing values:
4
2
***********MENU************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:8

Printing values:
4
2
```

```
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:8

Printing values:
4
2
***********MENU************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:5

Deleted node from last
***********MENU************
1.Insert in begining
2.Insert at last
3.Inseert at random location
4.Delete from begining
5.Delete from last
6.Delete at random location
7.Search for element
8.Display
9.Exit
Enter your choice:9


...Program finished with exit code 0
Press ENTER to exit console.
```

Submitted by: Gelle Hruthesh Reddy(20BCB7031)