Experiment 5.1
Date 05-11-2022

LINKED LIST OPERATIONS

Aim

Write a menu driven program for performing the following operations on a linked list
1. Display
2. Insert at beginning
3. Insert at end
4. Insert at a specified position
5. Delete from beginning
6. Delete from end
7. Delete from a specified position

Algorithm

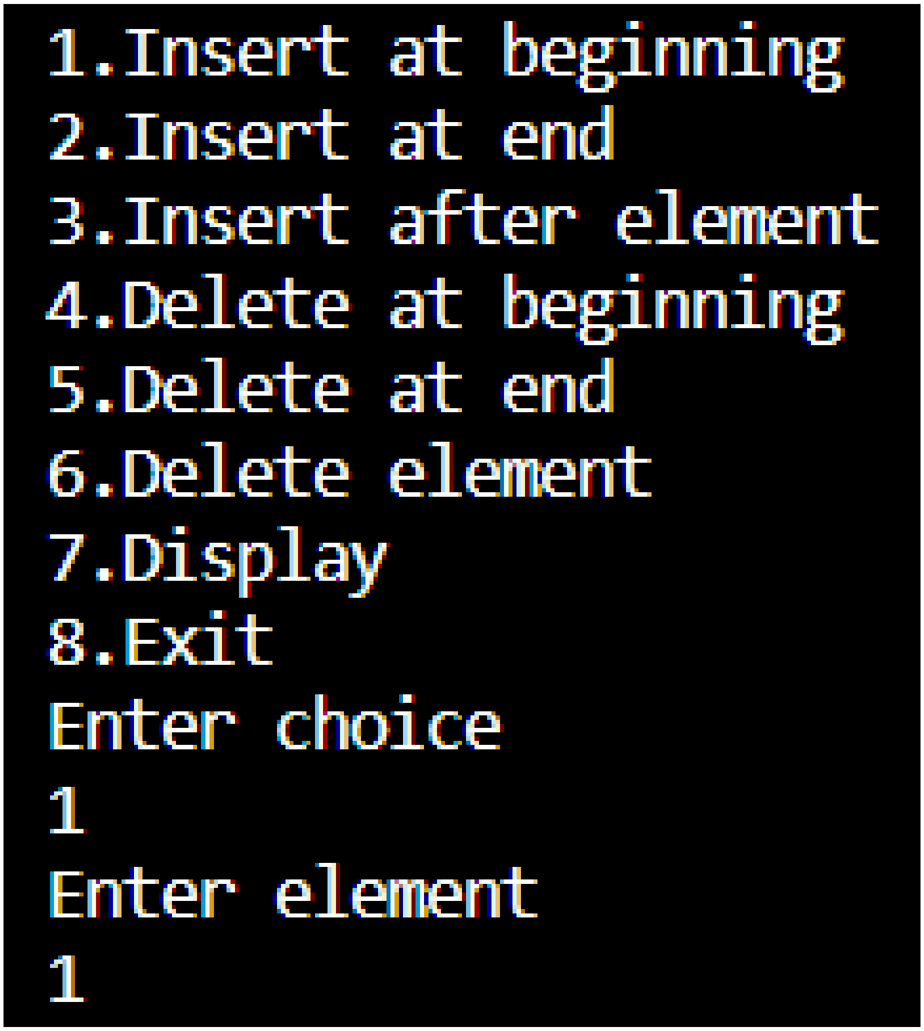
```
Let ptr be a pointer of struct node type and header be the beginning node.
In node there are two parts, data and link (which points next node).
GETNODE() - allocates new memory for node
ReturnNode() - frees the node space

Display()
    Step 1 : Start
    Step 2 : ptr = header -> link
    Step 3 : while( ptr != NULL)
    Step 3.1 : Print ptr -> data
    Step 3.2 : ptr -> link
    Step 4 : Stop

InsertFirst(item)
    Step 1 : Start
    Step 2 : new = GETNODE(node)
    Step 3 : if( new = NULL)
    Step 3.1 : Print "Memory underflow"
    Step 4 : else
    Step 4.1 : new -> link = header -> link
    Step 4.2 : new -> data = item
    Step 4.3 : header -> link = new
    Step 5 : Stop

InsertLast(item)
    Step 1 : Start
    Step 2 : new = GETNODE(node)
    Step 3 : if( new = NULL)
    Step 3.1 : Print "Memory underflow"
    Step 4 : else
    Step 4.1 : ptr = header
    Step 4.2 : while(ptr -> link != NULL)
    ptr = ptr -> link
    end while
    Step 4.3 : new -> link = ptr -> link
    Step 4.3 : new -> data = item
    Step 4.4 : ptr -> link = new
    Step 5 : Stop

InsertAny(item,key)
    Step 1 : Start
    Step 2 : new = GETNODE(node)
    Step 3 : if( new = NULL)
    Step 3.1 : Print "Memory underflow"
    Step 4 : else
    Step 4.1 : ptr = header
    Step 4.2 : while(ptr -> data != key)
    ptr = ptr -> link
    end while
    Step 4.3 : new -> link = ptr -> link
    Step 4.3 : new -> data = item
    Step 4.4 : ptr -> link = new
    Step 5 : Stop

DeleteFirst()
    Step 1 : Start
    Step 2 : ptr = header -> link
    Step 3 : if( ptr = NULL)
    Step 3.1 : Print "List empty"
    Step 4 : else
    Step 4.1 : header -> link = ptr -> link
    Step 4.2 : ReturnNode(ptr)
    Step 5 : Stop

DeleteLast()
    Step 1 : Start
    Step 2 : ptr = header -> link
    Step 3 : if( ptr = NULL)
    Step 3.1 : Print "List empty"
    Step 4 : else
    Step 4.1 : while(ptr -> link != NULL)
    temp = ptr
    ptr = ptr -> link
    end while
    Step 4.2 : temp -> link = NULL
    Step 4.3 : ReturnNode(ptr)
    Step 5 : Stop

DeleteAny(key)
    Step 1 : Start
    Step 2 : ptr = header -> link
    Step 3 : if( ptr = NULL)
    Step 3.1 : Print "List empty"
    Step 4 : else
    Step 4.1 : while(ptr -> data != key)
    temp = ptr
    ptr = ptr -> link
    end while
    Step 4.2 : temp -> link = NULL
    Step 4.3 : ReturnNode(ptr)
    Step 5 : Stop
```

Program code

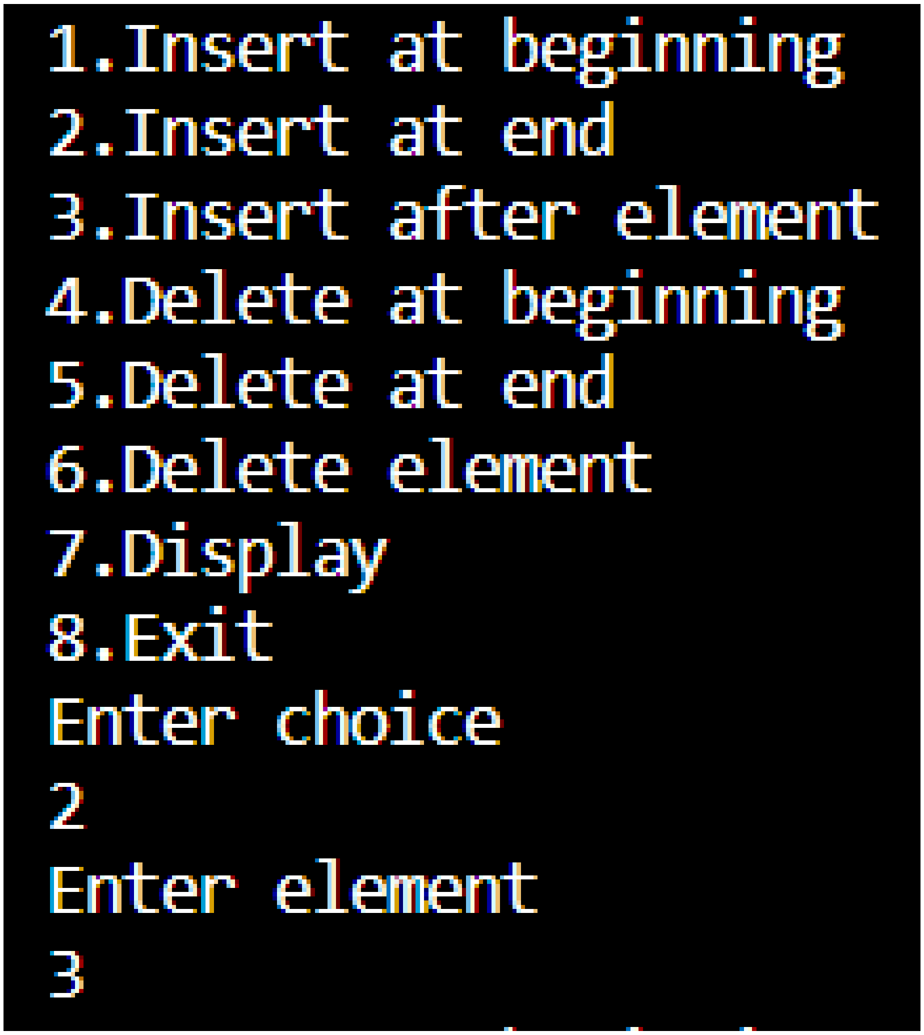
```
 #include<stdio.h>
#include<stdlib.h>
struct ptr
{
    int data;
    struct ptr*link;
};
struct ptr*head;
void insert_beg(int e)
{
    struct ptr*newptr=(struct ptr*)malloc(sizeof(struct ptr));
    if(head==NULL)
    {
        newptr->data=e;
        newptr->link=NULL;
        head=newptr;
    }
    else
    {
        newptr->data=e;
        newptr->link=head;
        head=newptr;
    }
}
void insert_end(int e)
{
    struct ptr*newptr=(struct ptr*)malloc(sizeof(struct ptr));
    struct ptr*temp=(struct ptr*)malloc(sizeof(struct ptr));
    if(head==NULL)
    {
        newptr->data=e;
        newptr->link=NULL;
        head=newptr;
    }
    else
    {
        newptr=head;
        while(newptr->link!=NULL)
            newptr=newptr->link;
        temp->data=e;
        temp->link=NULL;
        newptr->link=temp;
    }
}
void insert_mid(int e)
{
    int i,k=0;
    printf("Element to be inserted\n");
    scanf("%d",&i);
    struct ptr*newptr=(struct ptr*)malloc(sizeof(struct ptr));
    struct ptr*temp=(struct ptr*)malloc(sizeof(struct ptr));
    newptr=head;
    if(head==NULL)
    {
        newptr->data=e;
        newptr->link=NULL;
        head=newptr;
    }
    else
    {
        while(newptr->data!=e)
        {
            newptr=newptr->link;
            if(newptr==NULL)
            {
                k=1;
                break;
            }
        }
        if(k==0)
        {
            temp->data=i;
            temp->link=newptr->link;
            newptr->link=temp;
        }
        else
            printf("Invalid element\n")    ;
    }
}
void delete_beg()
{
    int e;
    if(head==NULL)
        printf("Empty list\n");
    else
    {
        e=head->data;
        head=head->link;
    }
    printf("Deleted is %d\n",e);
}
void delete_end()
{
    int e;
    struct ptr*temp1;
    struct ptr*temp2=(struct ptr*)malloc(sizeof(struct ptr));
    if(head==NULL)
        printf("Empty list\n");
    else
    {
        temp1=head;
        if(temp1->link==NULL)
        {
            e=temp1->data;
            head==NULL;
            free(temp1);
        }
        else
        {
            while(temp1->link!=NULL)
            {
                temp2=temp1;
                temp1=temp1->link;
            }
            e=temp1->data;
            temp2->link=NULL;
            free(temp1);
        }
    }
}
void delete_mid(int e)
{
    struct ptr*temp1;
    struct ptr*temp2=(struct ptr*)malloc(sizeof(struct ptr));
    if(head==NULL)
        printf("Empty list\n");
    else
    {
        temp1=head;
        while(temp1->data!=e && temp1->link!=NULL)
        {
            temp2=temp1;
            temp1=temp1->link;
        }
        temp2->link=temp1->link;
        free(temp1);
    }
}
void display()
{
    int i;
    struct ptr*temp;
    temp=head;
    if(temp==NULL)
        printf("Empty list\n");
    else
    {
        printf("List is \n");
        while(temp!=NULL)
        {
            printf("%d ",temp->data);
            temp=temp->link;
        }
        printf("\n");
    }
```

```
}
void main()
{
    int i,c;
    char ch=’Y’;
    while(ch==’y’||ch==’Y’)
    {
        printf("1.Insert at beginning\n2.Insert at end\n3.Insert after element\n4.Delete at
        printf("Enter choice\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
            {
                printf("Enter element\n");
                scanf("%d",&i);
                insert_beg(i);
                break;
            }
            case 2:
            {
                printf("Enter element\n");
                scanf("%d",&i);
                insert_end(i);
                break;
            }
            case 3:
            {
                printf("Enter elment to be inserted after\n");
                scanf("%d",&i);
                insert_mid(i);
                break;
            }
            case 4:
            {
                delete_beg();
                break;
            }
            case 5:
            {
                delete_end();
                break;
            }
            case 6:
            {
                printf("Enter element to be deleted\n");
                scanf("%d",&i);
                delete_mid(i);
                break;
            }
            case 7:
            {
                display();
                break;
            }
            default:
            {
                exit(0);
            }
        }
    }
}
```

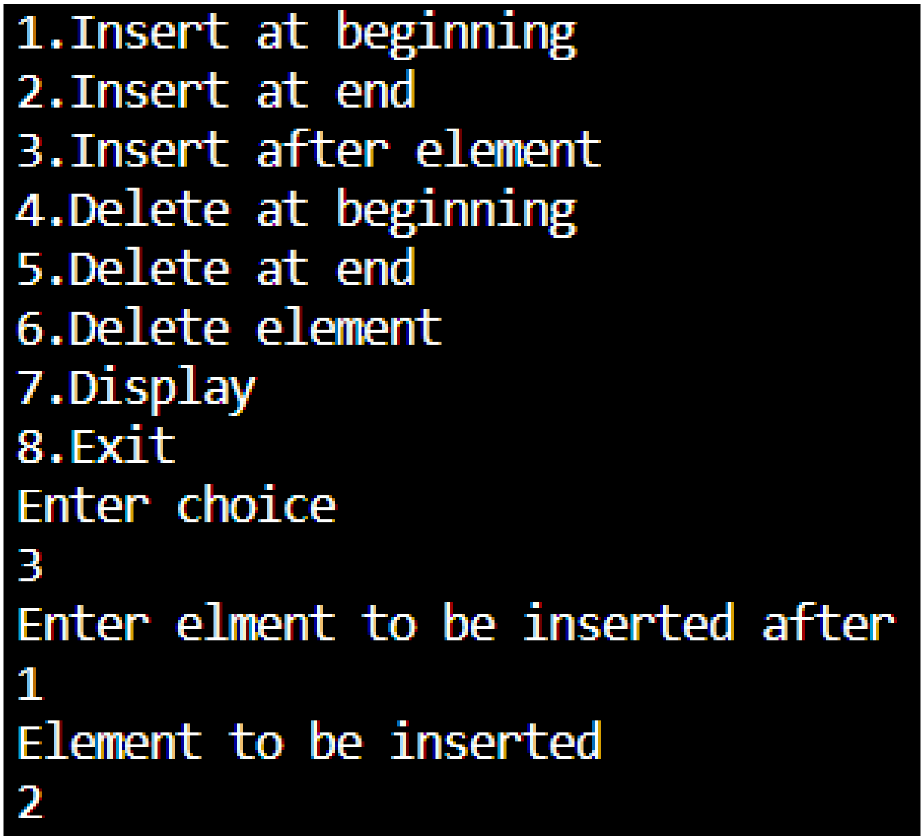
Sample Input and Output



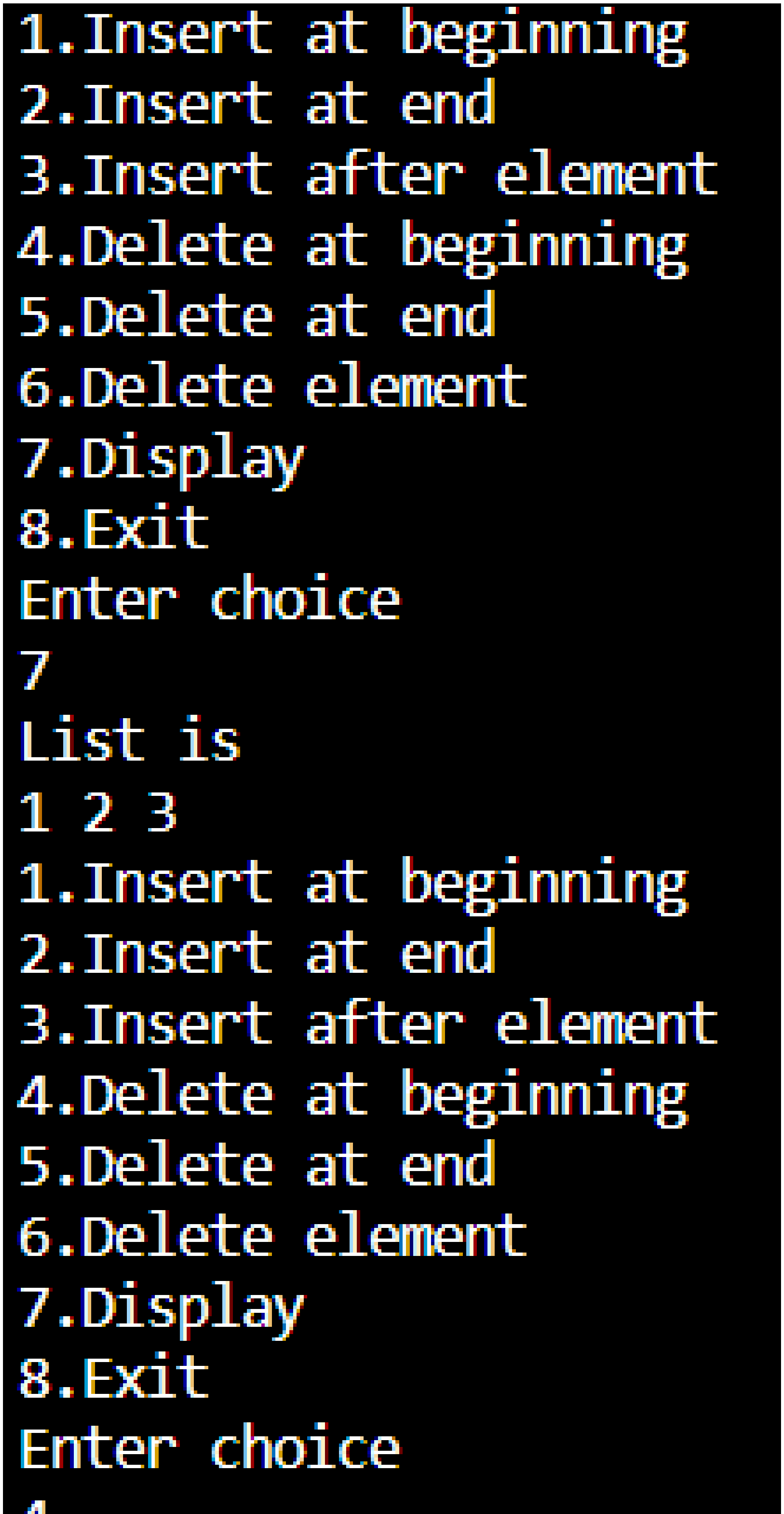
```
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
1
Enter element
1
```



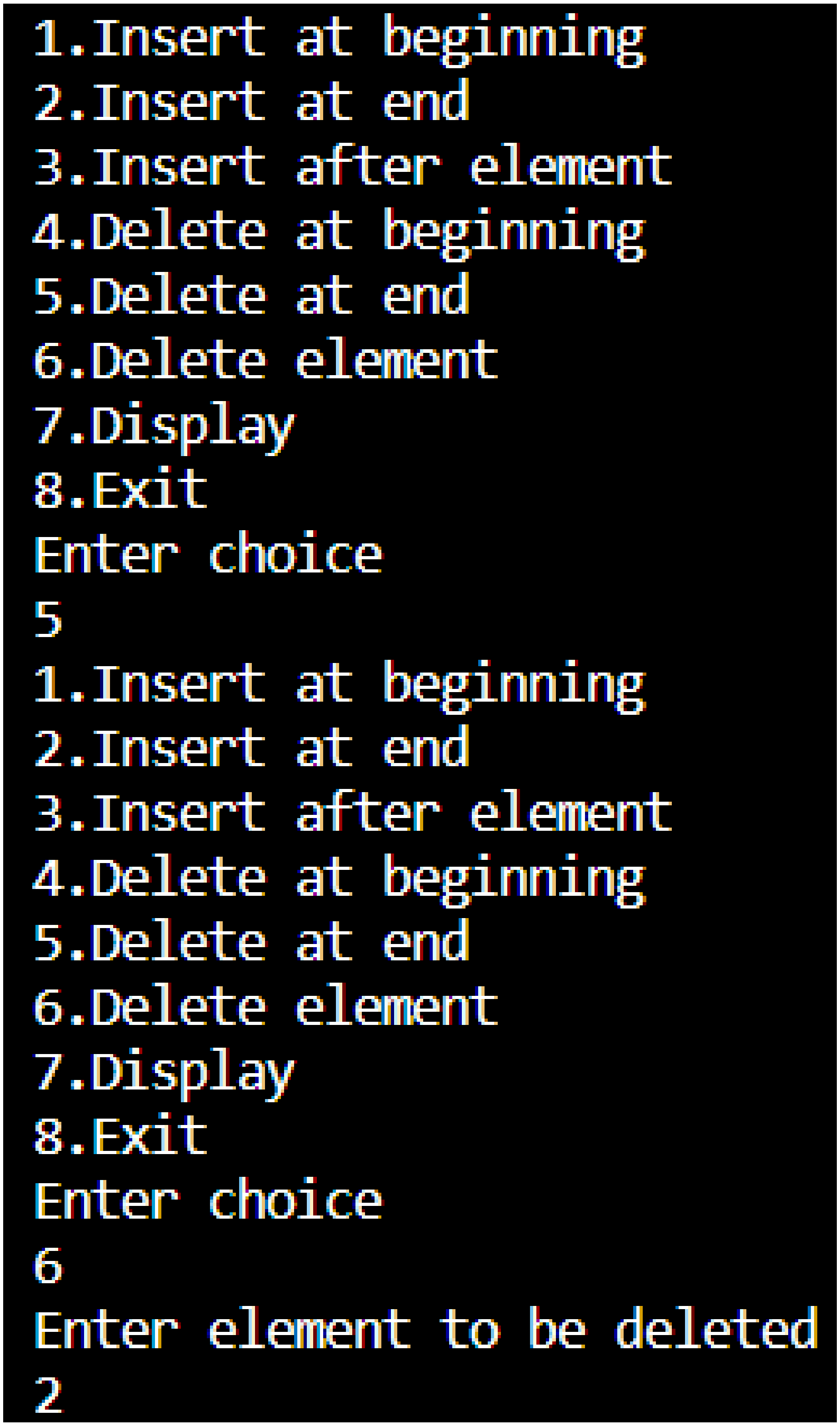
```
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
2
Enter element
3
```



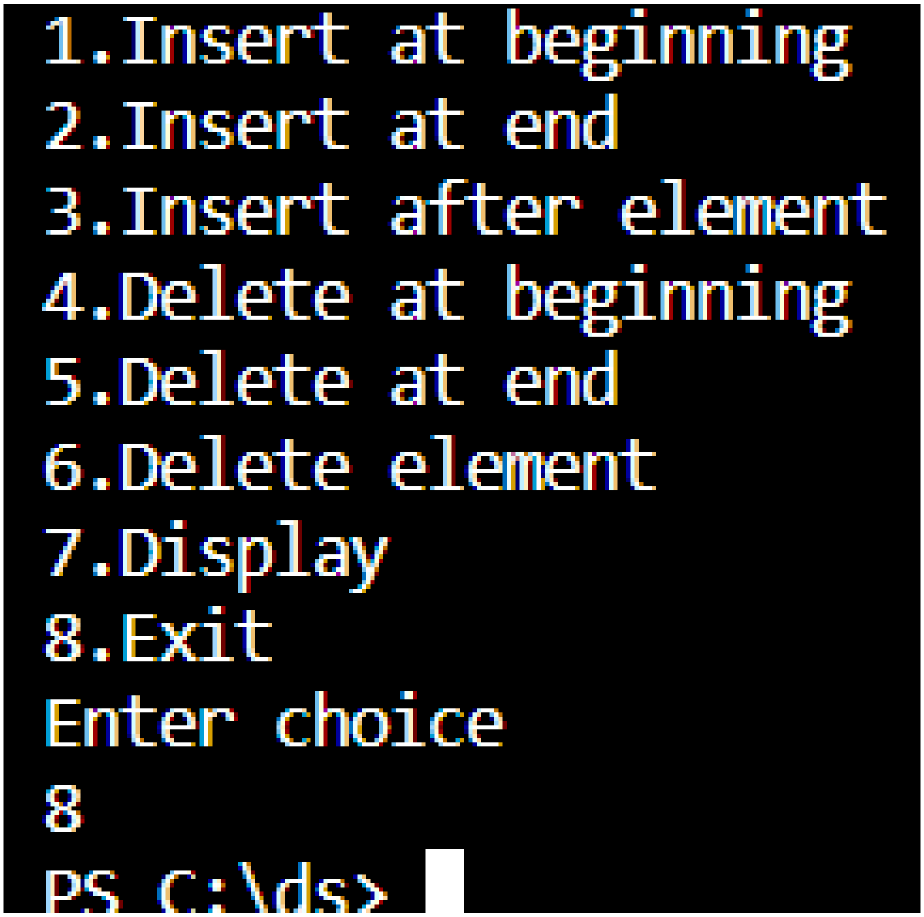
```
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
3
Enter elment to be inserted after
1
Element to be inserted
2
```



```
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
7
List is
1 2 3
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
4
```



```
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
5
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
6
Enter element to be deleted
2
```



```
1.Insert at beginning
2.Insert at end
3.Insert after element
4.Delete at beginning
5.Delete at end
6.Delete element
7.Display
8.Exit
Enter choice
8
PS C:\ds>
```

Output

Result

Program executed successfully