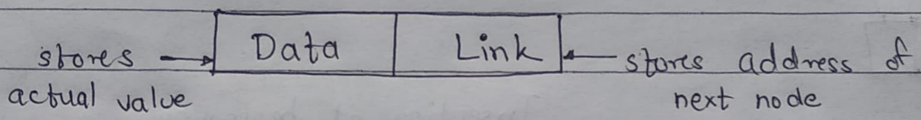
Shreyas Sawant
D7A 55

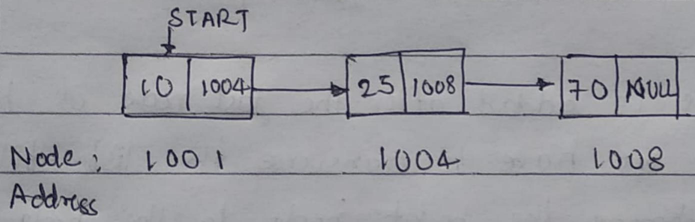
AIM: Single Linked List Array Implementation

THEORY:

- Single Linked List

It is a sequence of elements in which every element has link to its next element in the sequence in any single linked list, the individual elements is called as "Node". Every "Node" contains two fields, data field and the next field. The data field ~~&~~ is used to store value of node and next field to store address of next node in the sequence.





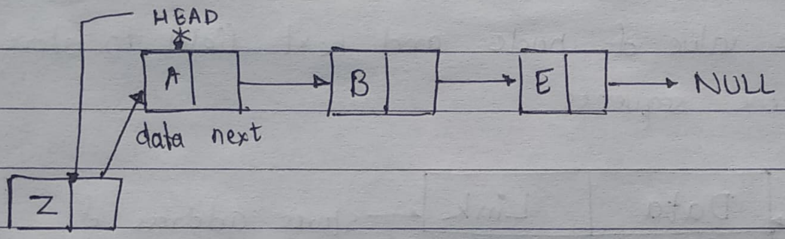
- Operations on Linked List

1) Insertion
2) Deletion
3) Display

1) Insertion :- Insertion can be performed in three ways :

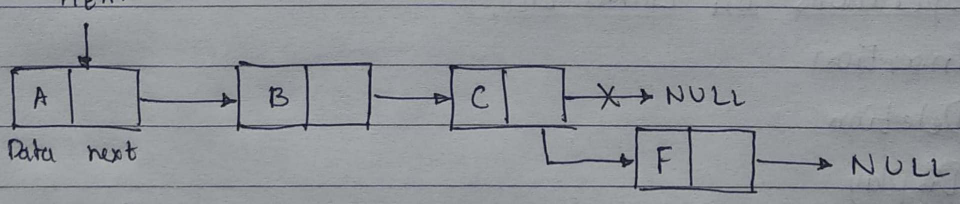
- At beginning
- At end
- At a specific position

a) At beginnig - The new node is always added before the head of the given list, and then the namely added node becomes the new head of the linked list



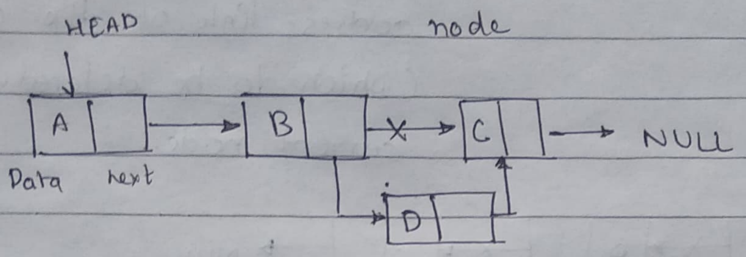
Insertion at beginning

b) At end - The new node is added after the last node of the given linked list, we have to transverse the list till the end and then change the next node to the node which is getting added.



Insertion at end

c). At a specific position: We are given a pointer to a node, and the new node is inserted after the given node

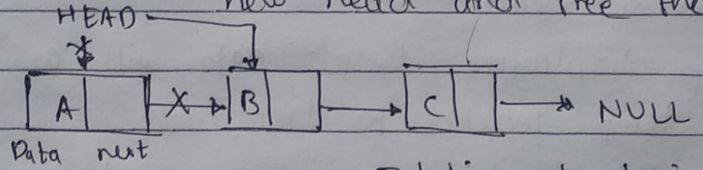


Insertion at specific location

2) Deletion: Delection can be performed in 3 ways:

- At beginning
- At end
- At a specific position

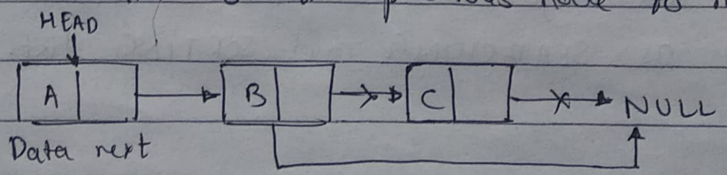
a) At beginning - Assign the next node after the head node to the new head and free the previous head



Deletion at beginning

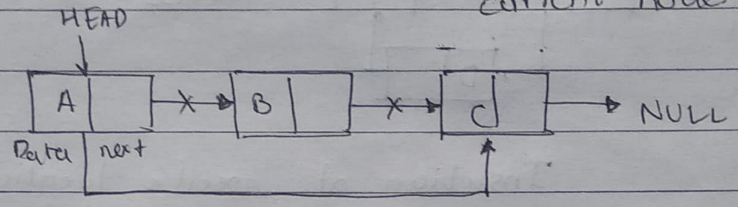
~~b)~~

b) At end - Travers the list and find the last node, assign the address link of the previous node to null and free the last node.



Deletion at end

c) Deletion at a specific position - Search node to be deleted, assign the address link of previous node to the address link of the canent node (which to be deleted) and then free current node



Deletion at a specific position

3) Display: ~~To~~ display the elenents of linked list, check whether the linkd list empty or not

- If empty, display as "empty list".
- If not empty, define a node pointer 'temp' and intialize 'head'.
- Keep displaying temp -> data with an arrow (->) until the ~~th~~ temp reaches to last node
- Finally display temp -> data with arrow pointing to Null (temp -> data -> NULL)

Above concepts and algorithms can also be used to perform operations such as SEARCHING and SORTING list.

SDS Page No.
Date / /

- ## Limitations of Single Linkd list:

1) Memory usage : More memory is required in the linkd list as compared to an array, as address storage of next node is required in each node.

2) Traversal - Traversal is more time consuming than an array.

3) Reverse Traversing - In a single linked list reverse traversing is not possible

4) Random access - Random access is not possible due to its dynamic memory allocation.

- Difference between Arrays and Linkd List

Arrays	Linked List
1) An array is a collection of elements of a similar data type	1) A linked list is a collection of object known as node where node consists of two parts: data and address
2) Array elements are stored in contiguous manner.	2) Linked list elements can be stored anywhere in the memory or randomly stored.
3) Array takes more time while perfoming operation like insertion, deletion etc.	3). Linked list takes less time while perfoming aperation like insertion, deletion etc.
4) In case of array memory is ~~at~~ allocated at compile time	4) In case of linked list memory is allocated at runtime.
5) Array works with static memory that is its size is fixed and cannot be changed at runtime	5) It works with dynamic memory and thus can be changed at runtime
6) Accessing element in array is faster as it can be directly accessed through the index.	6) Accessing element is slower as it needs to traverse the whole list to find the element.

CONCLUSION:

Errors encountered:

1) Accessing members of structure using dot (.) operater

```
temp.data
temp.next
```

Solution Using -> solves the error

```
temp->data
temp->next
```

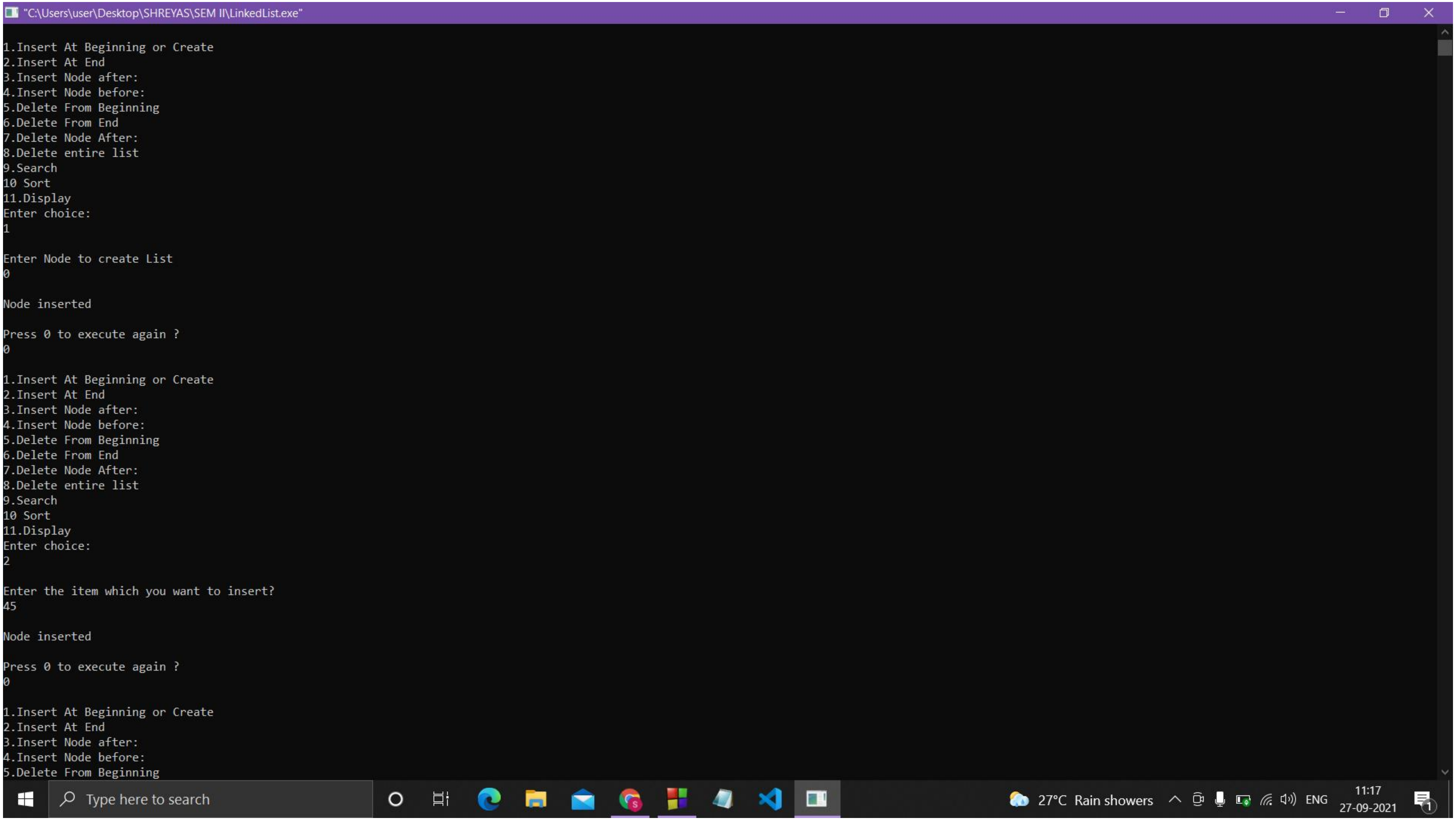
2) Missing semicolon after closing structure

```
Struct Node
{ int data; struct Node*next;
}
```

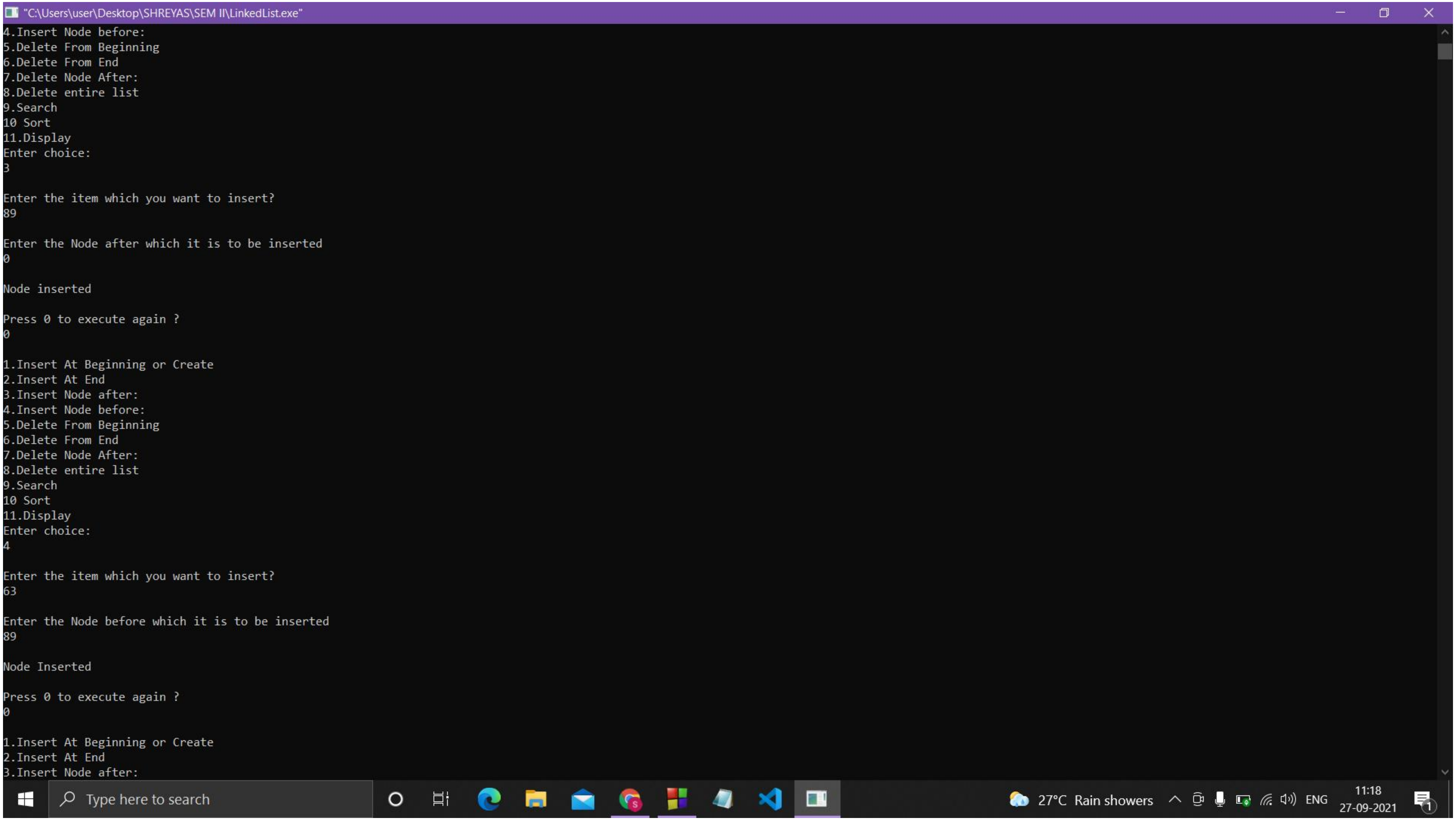
Solution Using correct syntax solves the error

```
struct Node
{ int data; struct Node *next;
};
```

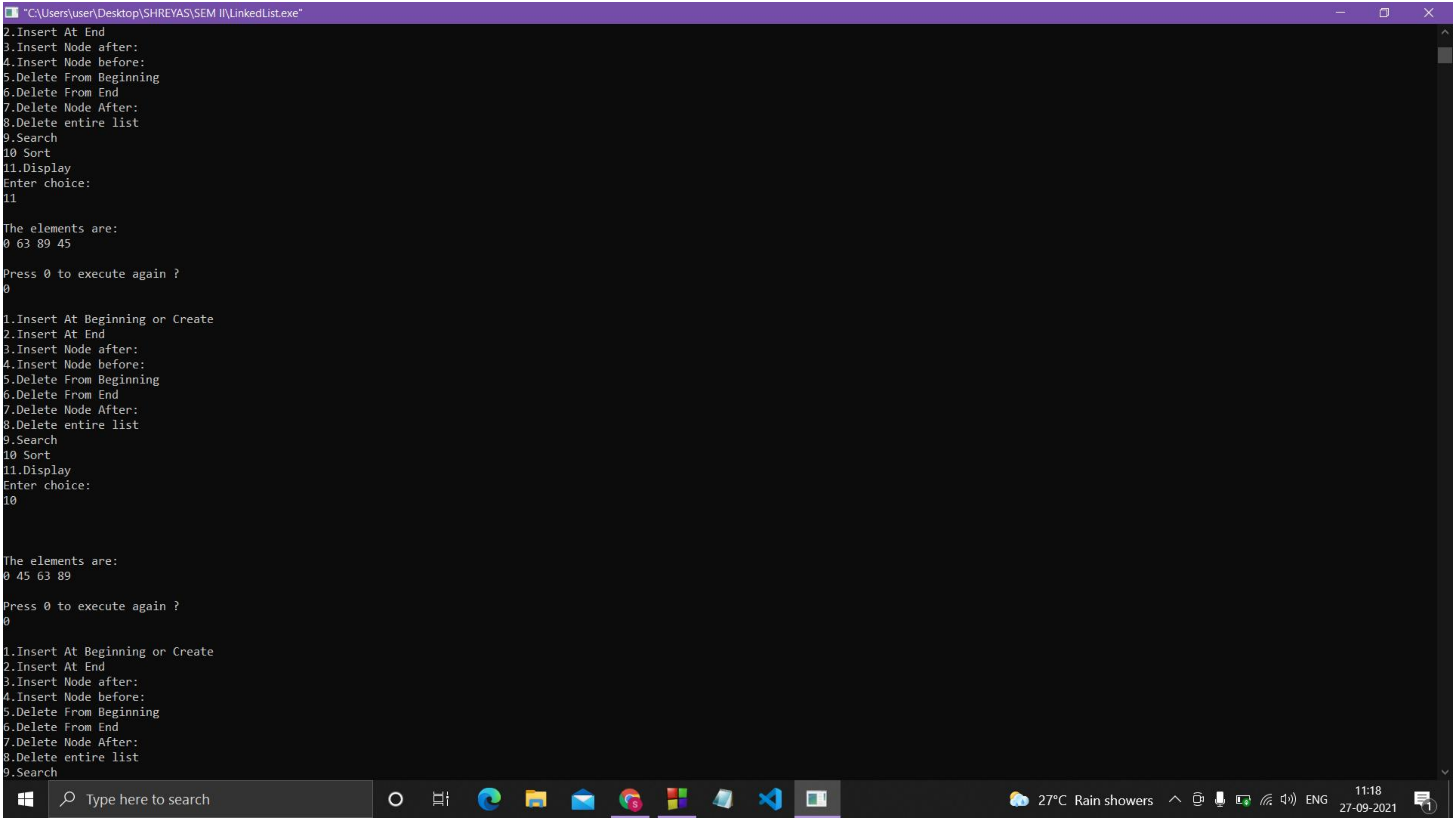
```c
#include <stdio.h>
#include <conio.h>
#include  <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
};
struct Node *head,*tail=NULL,*ptr,*temp;

void insAtBegin(int val)
{
    ptr =(struct Node *)malloc(sizeof(struct Node *));
    if(ptr == NULL)
        printf("\nOVERFLOW\n");
    else
    {   ptr->data=val;
        ptr->next = head;
        head=ptr;
        printf("\nNode inserted\n");
    }
}
void insAtEnd(int a)
{   ptr =(struct Node *)malloc(sizeof(struct Node *));
    temp=head;

    if(temp == NULL)
    {   head=ptr;
        ptr->data=a;
    ptr->next=NULL;
printf("\nNode inserted\n");
        return;
    }

        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        ptr->data=a;
        ptr->next=NULL;
        temp->next=ptr;

       printf("\nNode inserted\n");


}

void insAfter(int a,int b)
{   int k=0;
    ptr =(struct Node *)malloc(sizeof(struct Node *));
    temp =head;
    if(temp==NULL)
    {
        printf("\nEMPTY\n");return;
    }
    while(temp->next!=NULL)
    {
        if(temp->data==b)
        {
            k=1;break;
        }
        temp=temp->next;
    }

    if(k)
    {
        ptr->next=temp->next;
        ptr->data=a;
        temp->next=ptr;
        printf("\nNode inserted\n");
    }
    else
        printf("\nNOT FOUND\n");
}
void insBefore(int a,int b)
{   struct Node *pretemp;
    temp=head;
    ptr =(struct Node *)malloc(sizeof(struct Node *));
    pretemp =(struct Node *)malloc(sizeof(struct Node *));
    if(temp==NULL)
    {
        printf("\nEmpty\n");return;
    }
    ptr->data=a;

    while(temp->data!=b)
    {   pretemp=temp;
        temp=temp->next;
    }

    ptr->next=temp;
    pretemp->next=ptr;
    printf("\nNode Inserted\n");
    return;


}
void delBegin()
{temp=head;
    if(temp==NULL)
        {printf("\nEMPTY\n");return;}
    head=temp->next;
    temp->next=NULL;

    printf("\%d deleted\n",temp->data);
    free(temp);
}
void delEnd()
{ ptr =(struct Node *)malloc(sizeof(struct Node *));
    temp=head;

    if(temp==NULL)
        {printf("\nEmpty\n");return;}
    else if(head->next==NULL)
    {
        head=NULL;
        free(head);
        printf("Only Node deleted");
        return;
    }
    while(temp->next!=NULL)
    {   ptr=temp;
        temp=temp->next;
    }
    ptr->next=NULL;
    printf("\n%d Deleted\n",temp->data);
   free(temp);

}
void delAfter(int a)
{

    temp=head;

    if(temp==NULL)
       {

        printf("\nEmpty\n");return;}

    while(temp->data!=a)
    {
        temp=temp->next;
    }
    ptr=temp->next;
    temp->next=ptr->next;

    ptr->next=NULL;
    printf("\nNode deleted\n");
    free(ptr);
}
void deletelist()
{
    temp=head;
    if(temp==NULL)
    {
        printf("\nEmpty\n");return;
    }
    while(head->next!=NULL)
    {
        head=head->next;
        //head=head->next;
        temp->next=NULL;
    }head=NULL;
    printf("Deleted the whole list");return;
}
void sort()
{int k;printf("\n");
    temp=head;
    while(temp->next!=NULL)
    {
        ptr=temp->next;
        while(ptr!=NULL)
        {
            if(temp->data>ptr->data)
            {
                k=temp->data;
                temp->data=ptr->data;
                ptr->data=k;
            }ptr=ptr->next;
        }temp=temp->next;
    }
printf("\n");
}
void listsearch(int a)
{   ptr =(struct Node *)malloc(sizeof(struct Node *));
    temp=head;
    while(temp->next!=NULL)
    { ptr=temp->next;
        if(ptr->data==a||temp->data==a)
            {printf("\nFOUND\n");
            return;}
        else
            temp=temp->next;
    }
printf("\nNOT FOUND\n");
}
void display()
{ temp =head;
    if(temp==NULL)
    {
        printf("\nThe elements are:\nEMPTY\n");
    }
    else
    {   printf("\nThe elements are:\n");
        while(temp!=NULL)
        {printf("%d ",temp->data);
            temp=temp->next;
        }

    }printf("\n");
}

int main()
{
    int choice,item,k=0;
    do
    {   printf("\n1.Insert At Beginning or Create\n2.Insert At End\n3.Insert Node after:\n4.Insert Node before:\n5.Delete From Beginning\n6.Delete From End\n7.Delete Node After:\n8.Delete entire list\n9.Search\n10 Sort\n11.Display\nEnter choice:\n");
        int c;scanf("%d",&c);
        switch(c)
        {
            case 1:
        {   if(k==0)
                printf("\nEnter Node to create List\n");
            else
            printf("\nEnter the item which you want to insert?\n");
            scanf("%d",&item);k++;
            insAtBegin(item);break;
        }
            case 2:
                {
                    printf("\nEnter the item which you want to insert?\n");
                    scanf("%d",&item);
                    insAtEnd(item);break;
                }
            case 3:
                {   int n;
                    printf("\nEnter the item which you want to insert?\n");
                    scanf("%d",&item);
                    printf("\nEnter the Node after which it is to be inserted\n");
                    scanf("%d",&n);
                    insAfter(item,n);break;

                }
            case 4:
                {   int n;
                    printf("\nEnter the item which you want to insert?\n");
                    scanf("%d",&item);
                    printf("\nEnter the Node before which it is to be inserted\n");
                    scanf("%d",&n);
                    insBefore(item,n);break;
                }
            case 5:
                {
                    delBegin();break;

                }
            case 6:
                {
                    delEnd();break;
                }
            case 7:
                {int n;

                    printf("\nEnter the Node after which it is to be deleted\n");
                    scanf("%d",&n);
                    delAfter(n);break;
                }
            case 8:
                {
                    deletelist();break;
                }
            case 9:
                {   int n;
                    printf("\nEnter the Node to be searched\n");
                    scanf("%d",&n);
                    listsearch(n);break;
                }
            case 10:
                {
                    sort();

                }
            case 11:
                {
                    display();break;
                }
                default:printf("\nInvalid choice\n");
        }
        printf("\nPress 0 to execute again ?\n");
        scanf("%d",&choice);
    }while(choice == 0);

}
```



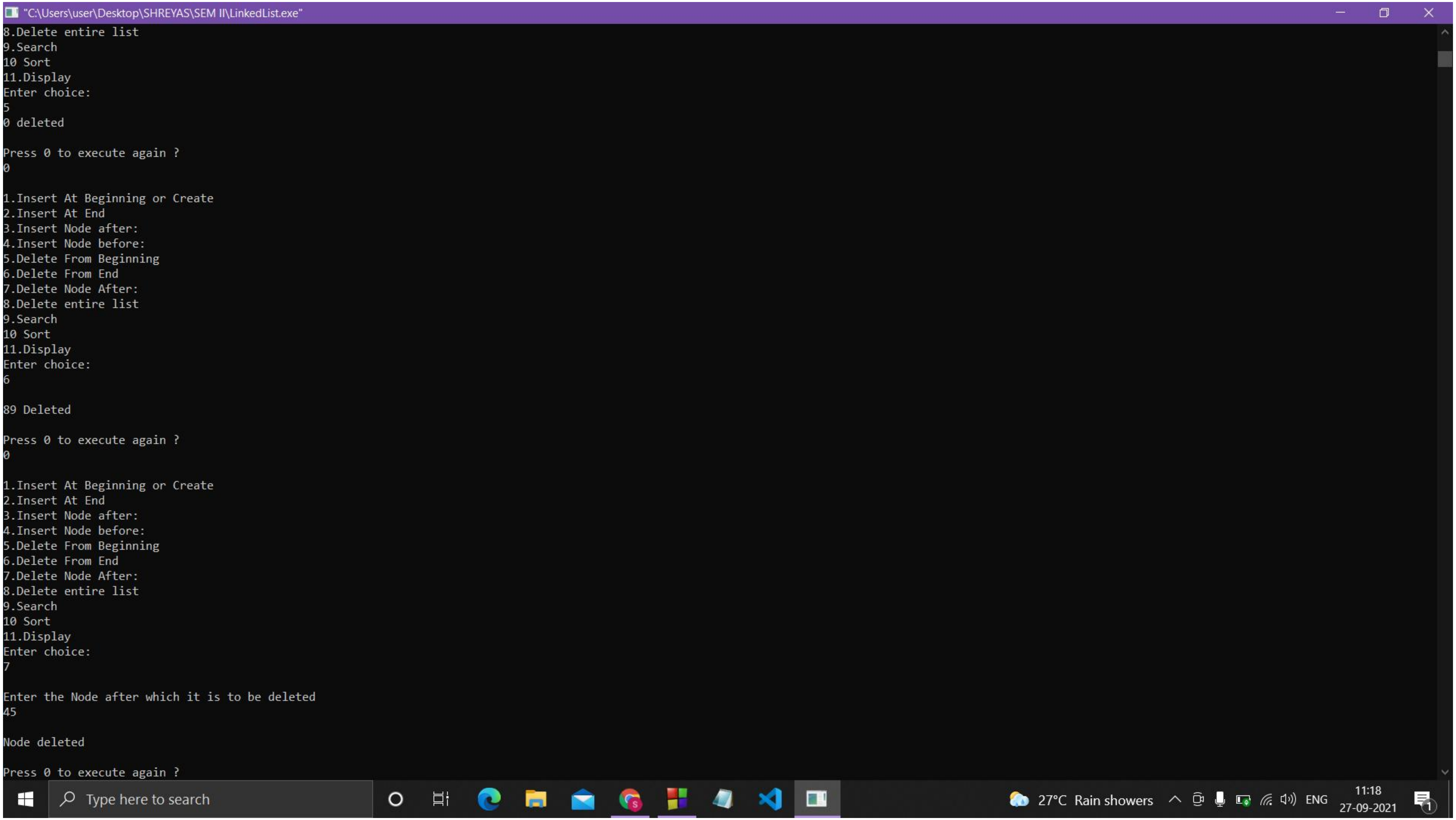
```
1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
1

Enter Node to create List
0

Node inserted

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
2

Enter the item which you want to insert?
45

Node inserted

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
```



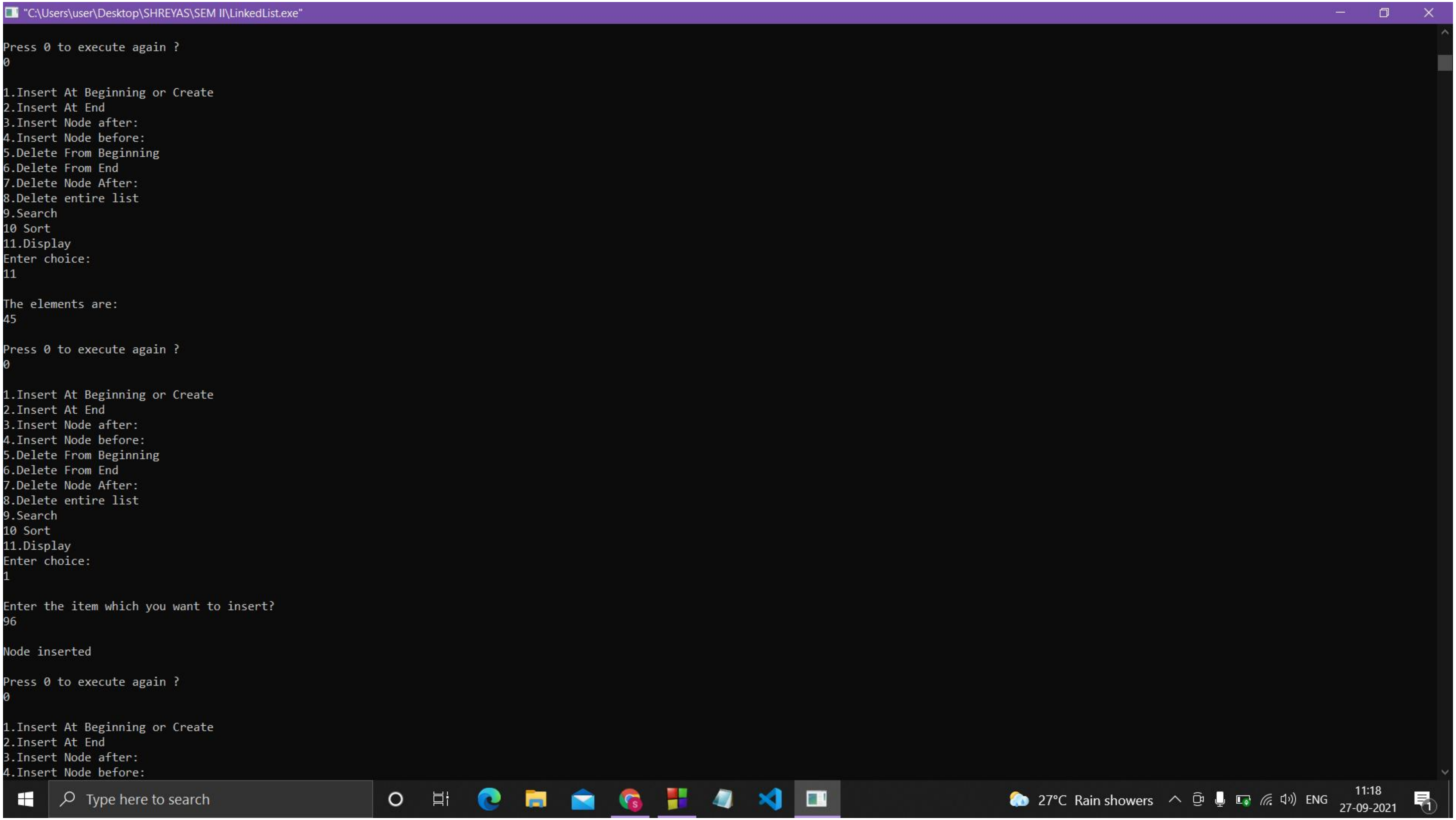
```
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
3

Enter the item which you want to insert?
89

Enter the Node after which it is to be inserted
0

Node inserted

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
4

Enter the item which you want to insert?
63

Enter the Node before which it is to be inserted
89

Node Inserted

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
```



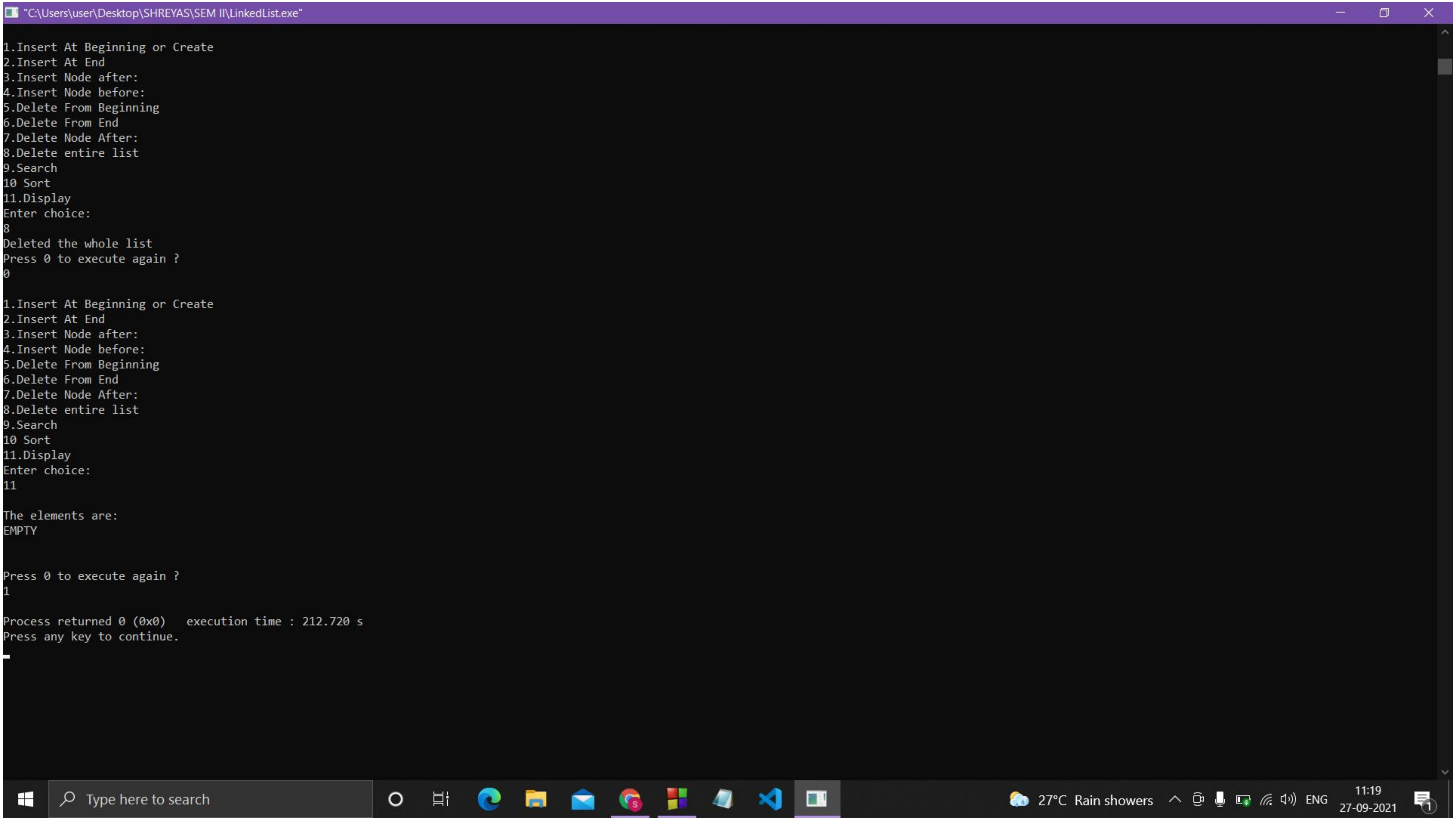
```
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
11

The elements are:
0 63 89 45

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
10



The elements are:
0 45 63 89

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
```



```
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
5
0 deleted

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
6

89 Deleted

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
7

Enter the Node after which it is to be deleted
45

Node deleted

Press 0 to execute again ?
```



```
Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
11

The elements are:
45

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
1

Enter the item which you want to insert?
96

Node inserted

Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
```



```
1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
8
Deleted the whole list
Press 0 to execute again ?
0

1.Insert At Beginning or Create
2.Insert At End
3.Insert Node after:
4.Insert Node before:
5.Delete From Beginning
6.Delete From End
7.Delete Node After:
8.Delete entire list
9.Search
10 Sort
11.Display
Enter choice:
11

The elements are:
EMPTY


Press 0 to execute again ?
1

Process returned 0 (0x0)   execution time : 212.720 s
Press any key to continue.
```