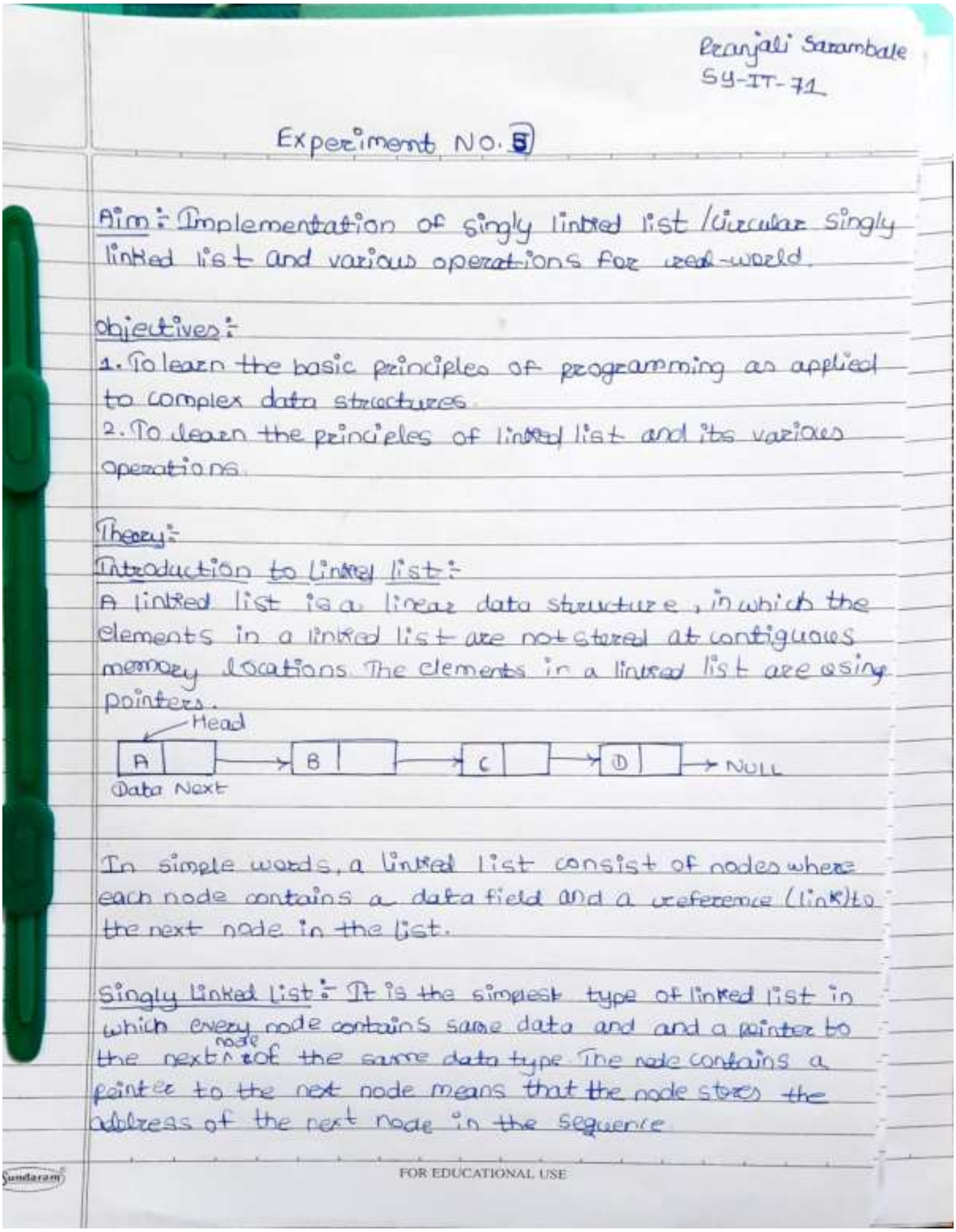


Pranjali Sarambale
SY-IT-71

Experiment No. 5

Aim: Implementation of singly linked list /Circular singly linked list and various operations for real-world.

Objectives:

1. To learn the basic principles of programming as applied to complex data structures.
2. To learn the principles of linked list and its various operations.

Theory:

Introduction to Linked list:

A linked list is a linear data structure, in which the elements in a linked list are not stored at contiguous memory locations. The elements in a linked list are using pointers.

Head
A → B → C → D → NULL
Data Next

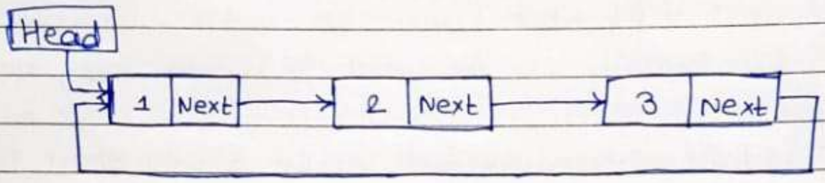
In simple words, a linked list consist of nodes where each node contains a data field and a reference (link) to the next node in the list.

Singly Linked List: It is the simplest type of linked list in which every node contains same data and and a pointer to the next node of the same data type. The node contains a pointer to the next node means that the node stores the address of the next node in the sequence.

Sundaram
FOR EDUCATIONAL USE

Introduction to circular linked list ~~th~~

In a circular singly linked list, the ~~dat~~ last node of the list contains a pointer to the first node of the list. we can contains a pointer to the circular singly linked list as well as circular doubly linked list. as well as circular doubly linked list. we traverse a circular linked list until we reach the same node where we started. The circular singly linked list has no beginning and no ending. There is no null value present in the next part of any of the nodes



Circular linked lists are mostly used in task maintainance in operating systems. There are many examples where circular linked list are being used in computer science including browser surfering where a record of pages visited in the past by the user, is maintained in the form of circular linked lists and can be accessed again on clicking the previous button.

Insertion :-

The insertion into a singly linked list can be performed at different positions. Based on the position of the new

FOR EDUCATIONAL USE

node being inserted, the insertion is categorized into the following categories :-

1) Insertion at beginning :- It involves inserting any element at the front of the list

2) Insertion at the end of the list :- It involves insertion at the last of linked list. The new node can be inserted as the only node in the list or it can be inserted as the last node.

3) Insertion after specified node :- It involves insertion after the specified node of the linked list. we need to skip the desired number of nodes in order to reach the node after which the new node will be inserted.

Deletion :-

The deletion of a node from a singly linked list can be performed at the different position. Based on the position of node being deleted, the operation is categorized as :-

1) Deletion at beginning :- It involves deletion of a node from the beginning of the list.

2) Deletion at end :- It involves deleting the last node of the list.

FOR EDUCATIONAL USE

→ Deletion after specified node : It involves deleting the node after the specified node in the list

Traversing :-

In traversing , we simply visit each node of the list at least once in order to perform some specific operation on it.

Algorithm :-

Insertion in the beginning

```
Step 1 : IF PTR = NULL
         Write OVERFLOW
          Go TO step 7
           [END OF IF]
Step 2 : SET NEW_NODE = PTR

Step 3 : SET PTR = PTR → NEXT
Step 4 : SET NEW-NODE → DATA = VAL
Step 5 : SET NEW = NODE → NEXT = HEAD
Step 6 : SET HEAD = NEW - NODE
Step 7 : Exit
```

Insertion at the End.

```
Step 1 : IF PTR = NULL Write Overflow
          GO to Step 1
          END OF IF
Step 2 : SET NEW - NODE = PTR
```

```
step 3: NEW_NODE → DATA = VAL
step 4: SET TEMP = HEAD
step 5: SET I = 0
step 6: REPEAT STEP 5 and 6 until 1
step 7: TEMP= TEMP → NEXT
step 8: IF TEMP = NULL
    WRITE "DESIRED NODE NOT PRESENT"
            GO TO STEP 12
            Find OF IF
            END OF LOOP
STEP 9: PTR → NEXT = TEMP → NEXT
STEP 10: TEMP → NEXT = PTR
STEP 11: SET PTR = NEW-NODE
STEP 12: EXIT
```

DELETION of beginning:

```
Step 1: IF HEAD = NULL
    WRITE UNDERFLOW
        Go to Step 5
        [END of IF]
Step 2: SET PTR = HEAD
Step 3: SET HEAD = HEAD → NEXT
Step 4: FREE PTR
Step 5 : EXIT
```

Deletion of specified node:

```
Step 1 : IF HEAD = NULL
```

```
        WRITE UNDERFLOW
        GOTO STEP 10
        END OF IF
Step 2 : SET TEMP = HEAD
Step 3 : SET HEAD = HEAD → NEXT
Step 4 : FREE PTR
Step 5 : EXIT
```

Deletion at specified node :

```
Step 1 : IF HEAD = NULL
      GOTO UNDERFLOW
      GOTO STEP 10
      END OF IF
Step2 : SET TEMP = HEAD
Step 3 : SET I = 0
Step4 : REPEAT STEP 5, TO 8 UNTIL, I
Step5 : TEMP 1 = TEMP
Step 6 : TEMP = TEMP → NEXT
Step 7 : IF TEMP = NULL
   WRITE "DESIRED NODE NOT PRESENT"
   GOTO STEP 12
   END OF IF
Step 8 : I = I + 1
   END OF LOOP
Step 9 : TEMP 1 → NEXT = TEMP → NEXT
Step 10 : FREE TEMP
Step 11 : EXIT
```

Deletion at the End:

```
Step 1: IF HEAD = NULL
Write UNDERFLOW
 Go to step 8
 [END OF IF]
Step 2: SET PTR = HEAD
Step 3: Repeat steps 4 and 5 while PTR→NEXT! =
NULL
Step 4: SET PREPTR = PTR
Step5: SET PTR = PTR→NEXT
 [END OF LOOP]
Step 6: SET PREPTR→NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT
```

Examples:

1) List of image that need to be burned to a CD in a medical, imaging, application.

2) List of users of website that need to be emailed some notification.

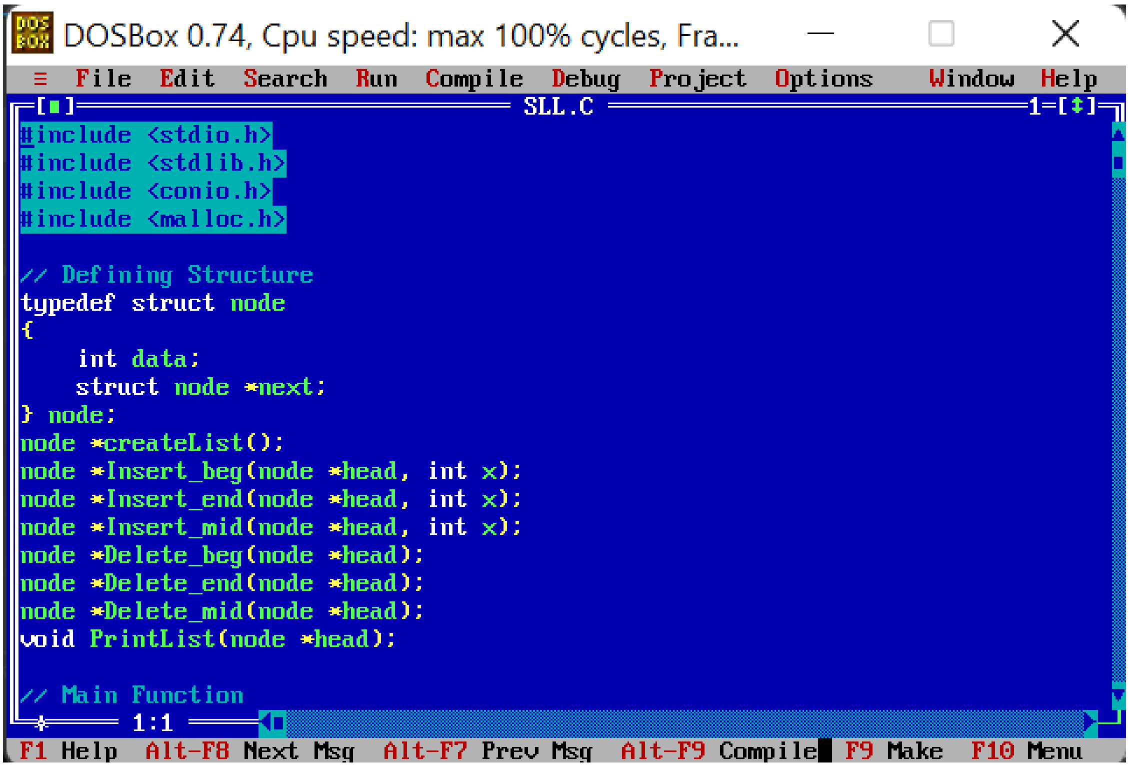
3) List of objects in a 3D game that need to be rendered to the screen

conclusion :- Thus, we have studied the concepts and implementations of singly linked list and its various operations.

Outcome :- Apply the concepts of singly linked list for real - world application.

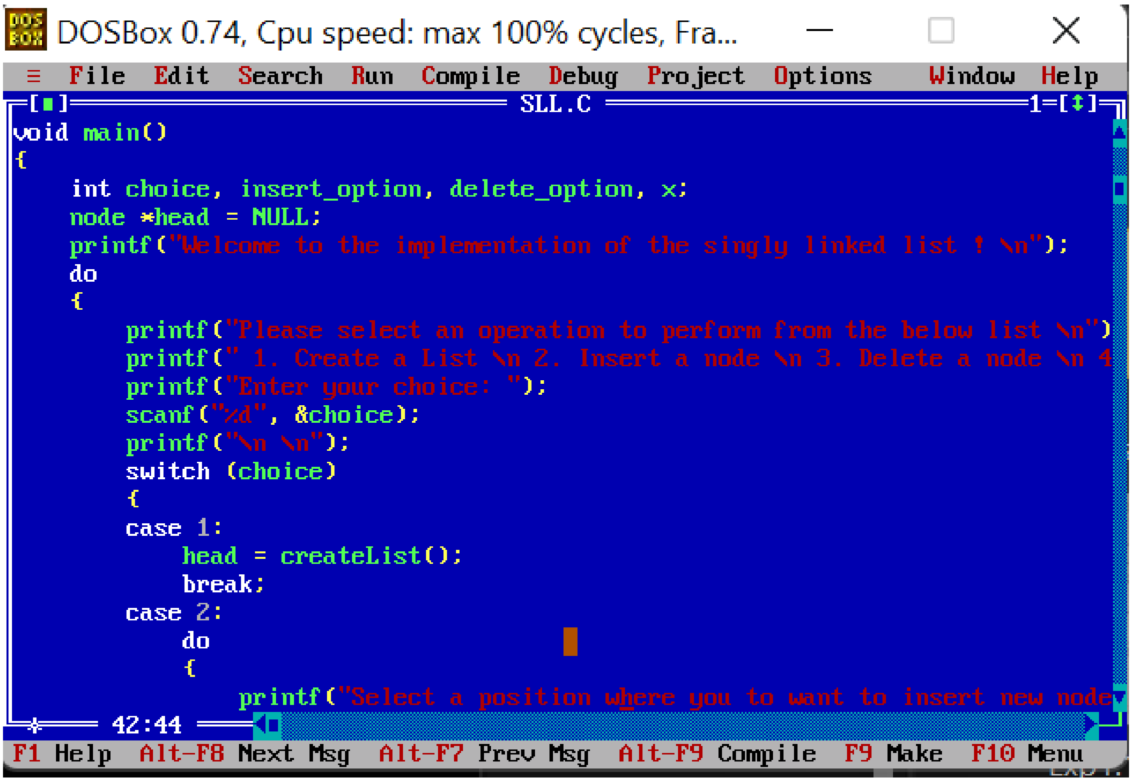
FOR EDUCATIONAL USE

PROGRAM: SINGLY LINKED LIST-SLL



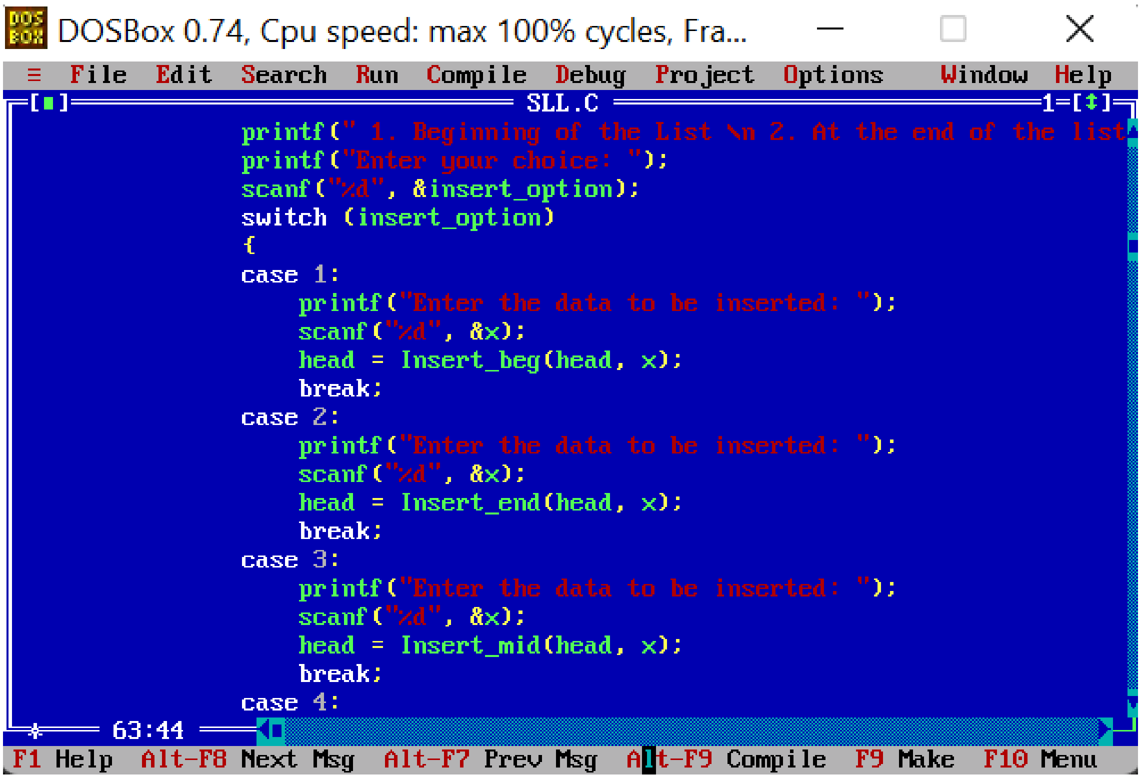
DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
SLL.C
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

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

// Defining Structure
typedef struct node
{
    int data;
    struct node *next;
} node;
node *createList();
node *Insert_beg(node *head, int x);
node *Insert_end(node *head, int x);
node *Insert_mid(node *head, int x);
node *Delete_beg(node *head);
node *Delete_end(node *head);
node *Delete_mid(node *head);
void PrintList(node *head);

// Main Function
```

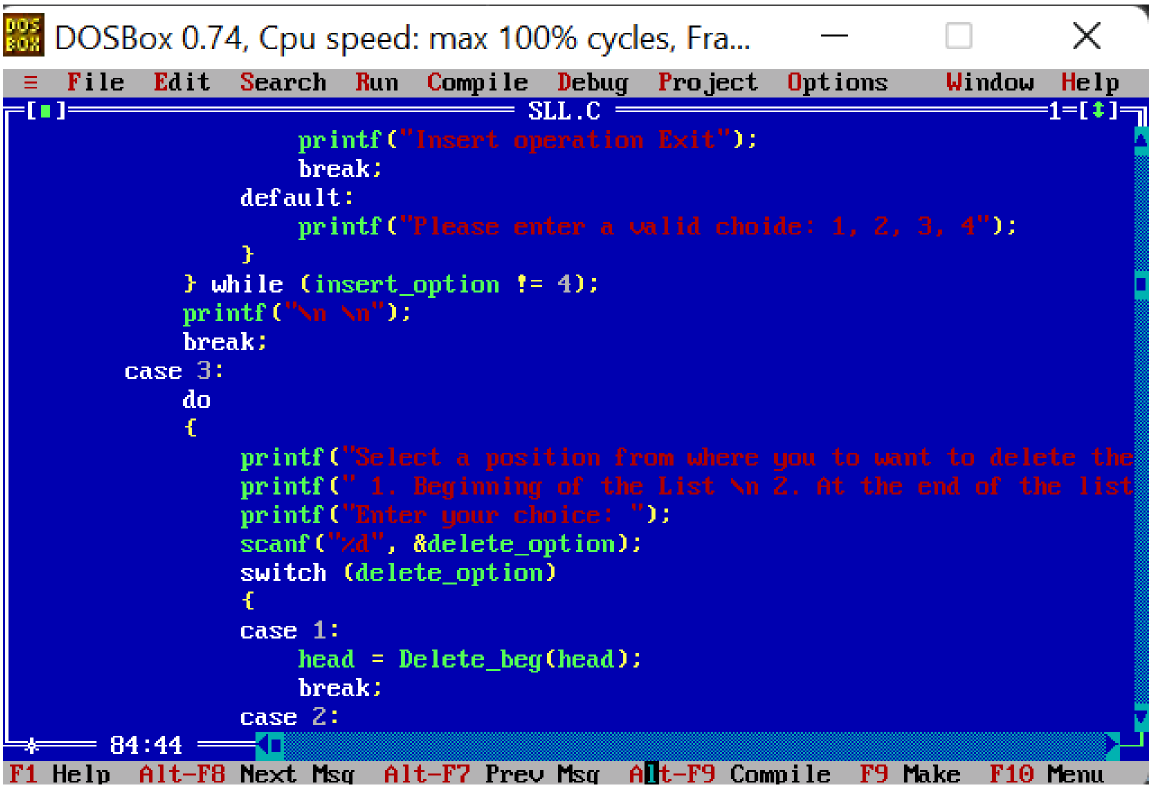


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
SLL.C
42:44
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

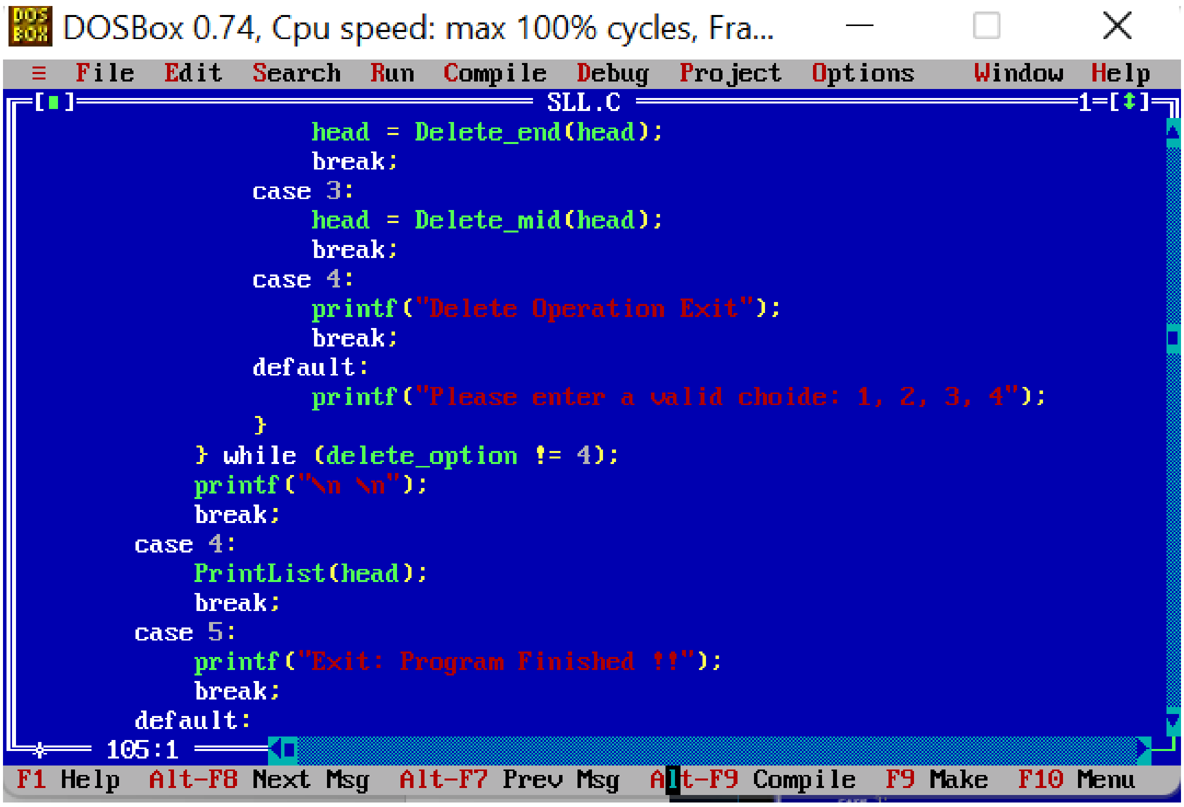
```
void main()
{
    int choice, insert_option, delete_option, x;
    node *head = NULL;
    printf("Welcome to the implementation of the singly linked list ! \n");
    do
    {
        printf("Please select an operation to perform from the below list \n")
        printf(" 1. Create a List \n 2. Insert a node \n 3. Delete a node \n 4
        printf("Enter your choice: ");
        scanf("%d", &choice);
        printf("\n \n");
        switch (choice)
        {
        case 1:
            head = createList();
            break;
        case 2:
            do
            {
                printf("Select a position where you to want to insert new node
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] SLL.C 1=[↕]
63:44
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

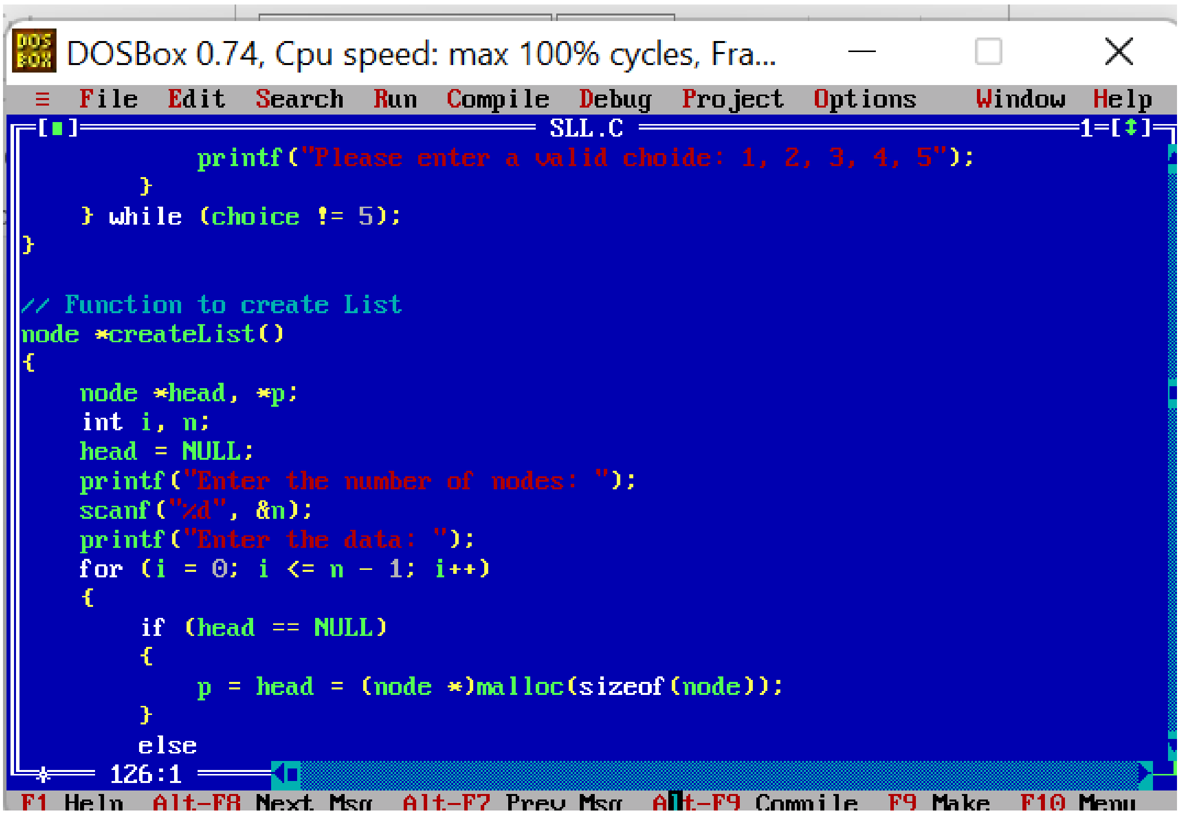
```
                printf(" 1. Beginning of the List \n 2. At the end of the list
                printf("Enter your choice: ");
                scanf("%d", &insert_option);
                switch (insert_option)
                {
                case 1:
                    printf("Enter the data to be inserted: ");
                    scanf("%d", &x);
                    head = Insert_beg(head, x);
                    break;
                case 2:
                    printf("Enter the data to be inserted: ");
                    scanf("%d", &x);
                    head = Insert_end(head, x);
                    break;
                case 3:
                    printf("Enter the data to be inserted: ");
                    scanf("%d", &x);
                    head = Insert_mid(head, x);
                    break;
                case 4:
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] SLL.C 1=[↕]
84:44
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

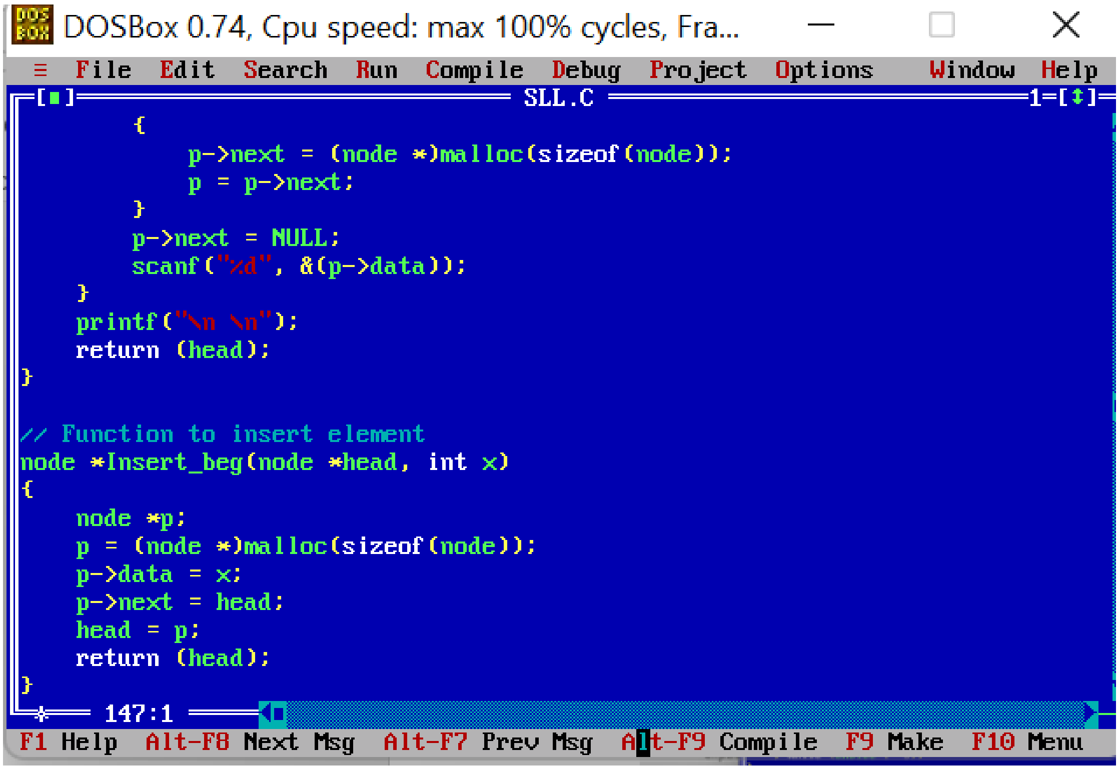
```
                    printf("Insert operation Exit");
                    break;
                default:
                    printf("Please enter a valid choide: 1, 2, 3, 4");
                }
            } while (insert_option != 4);
            printf("\n \n");
            break;
        case 3:
            do
            {
                printf("Select a position from where you to want to delete the
                printf(" 1. Beginning of the List \n 2. At the end of the list
                printf("Enter your choice: ");
                scanf("%d", &delete_option);
                switch (delete_option)
                {
                case 1:
                    head = Delete_beg(head);
                    break;
                case 2:
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
SLL.C
1
105:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

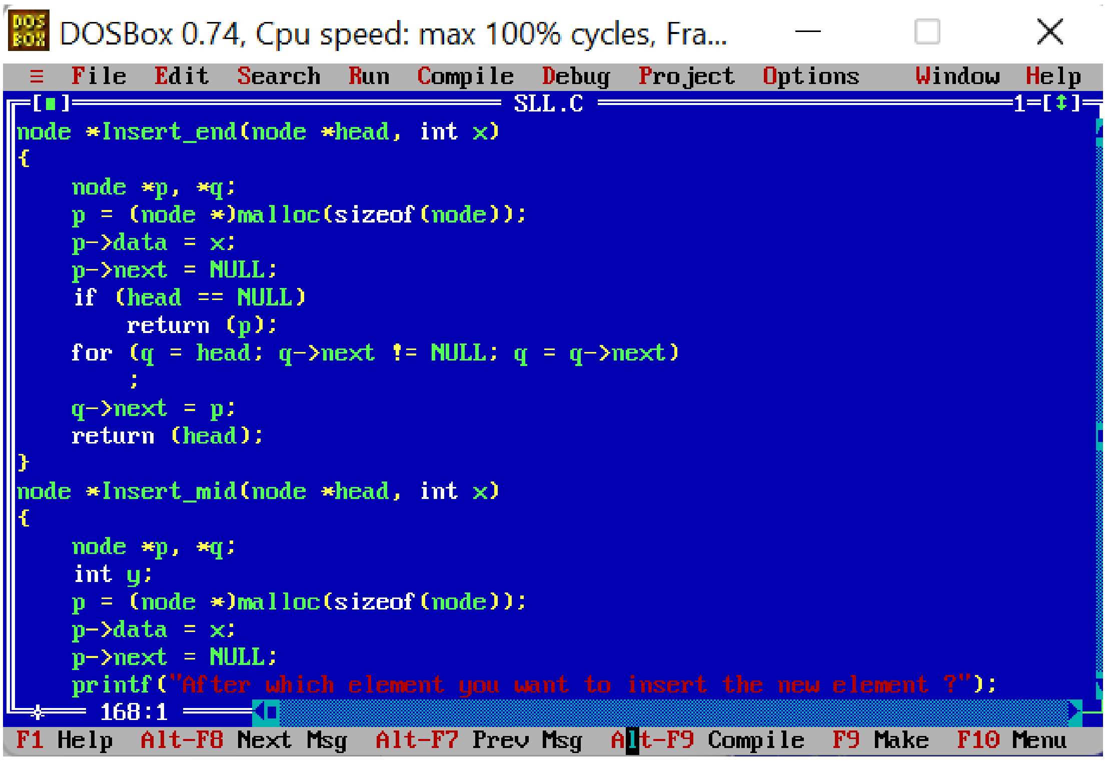
```
                    head = Delete_end(head);
                    break;
                case 3:
                    head = Delete_mid(head);
                    break;
                case 4:
                    printf("Delete Operation Exit");
                    break;
                default:
                    printf("Please enter a valid choide: 1, 2, 3, 4");
                }
            } while (delete_option != 4);
            printf("\n \n");
            break;
        case 4:
            PrintList(head);
            break;
        case 5:
            printf("Exit: Program Finished !!");
            break;
        default:
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
SLL.C
1
126:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

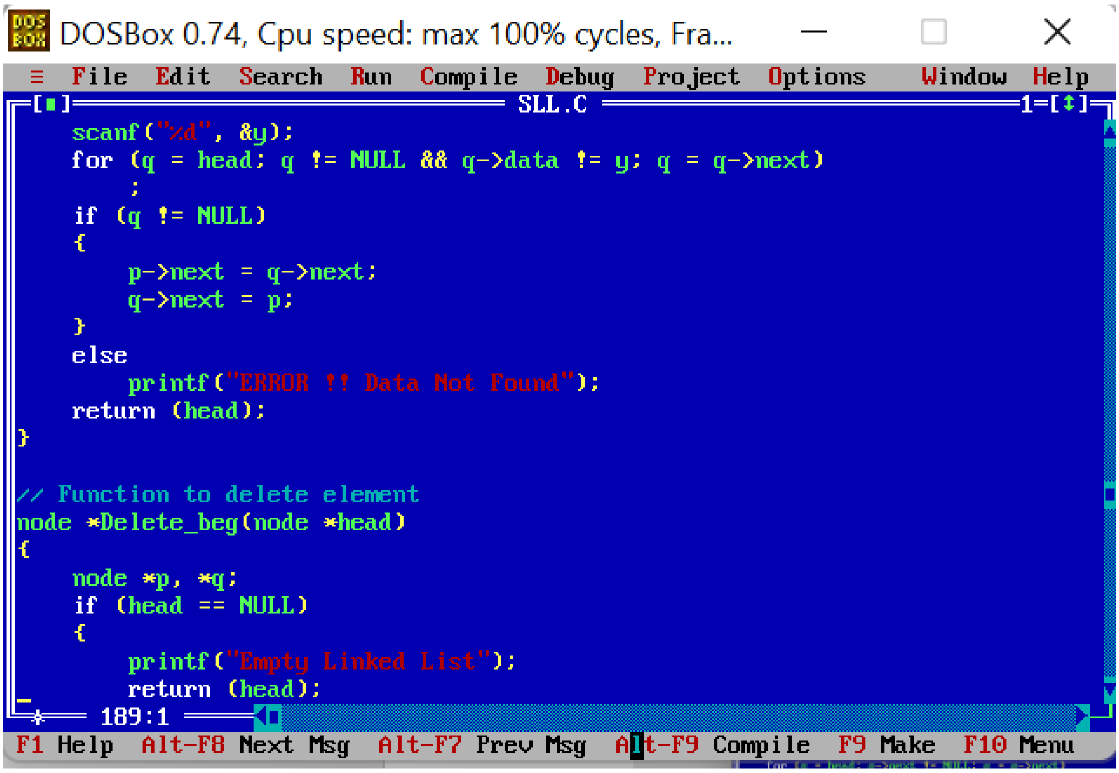
```
            printf("Please enter a valid choide: 1, 2, 3, 4, 5");
        }
    } while (choice != 5);
}

// Function to create List
node *createList()
{
    node *head, *p;
    int i, n;
    head = NULL;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("Enter the data: ");
    for (i = 0; i <= n - 1; i++)
    {
        if (head == NULL)
        {
            p = head = (node *)malloc(sizeof(node));
        }
        else
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
SLL.C
1
147:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

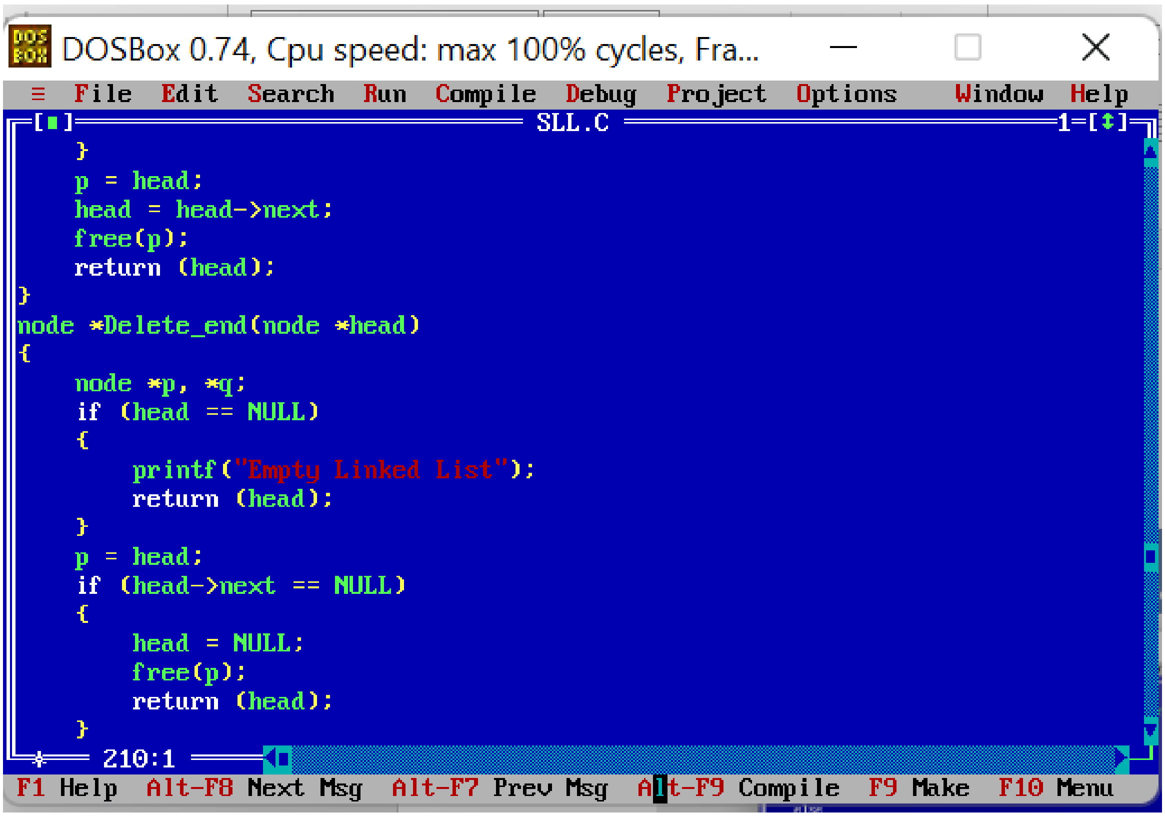
```
        {
            p->next = (node *)malloc(sizeof(node));
            p = p->next;
        }
        p->next = NULL;
        scanf("%d", &(p->data));
    }
    printf("\n \n");
    return (head);
}

// Function to insert element
node *Insert_beg(node *head, int x)
{
    node *p;
    p = (node *)malloc(sizeof(node));
    p->data = x;
    p->next = head;
    head = p;
    return (head);
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
SLL.C
1
168:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

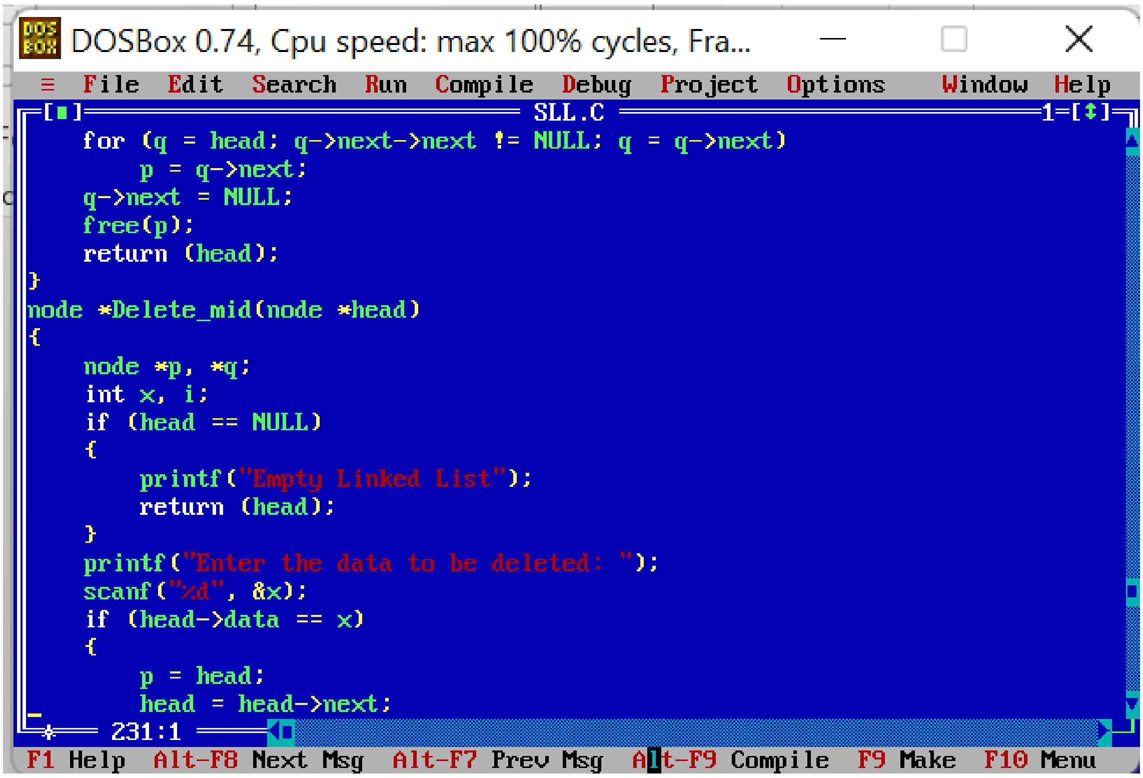
```
node *Insert_end(node *head, int x)
{
    node *p, *q;
    p = (node *)malloc(sizeof(node));
    p->data = x;
    p->next = NULL;
    if (head == NULL)
        return (p);
    for (q = head; q->next != NULL; q = q->next)
        ;
    q->next = p;
    return (head);
}
node *Insert_mid(node *head, int x)
{
    node *p, *q;
    int y;
    p = (node *)malloc(sizeof(node));
    p->data = x;
    p->next = NULL;
    printf("After which element you want to insert the new element ?");
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] SLL.C 1=[↕]
189:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

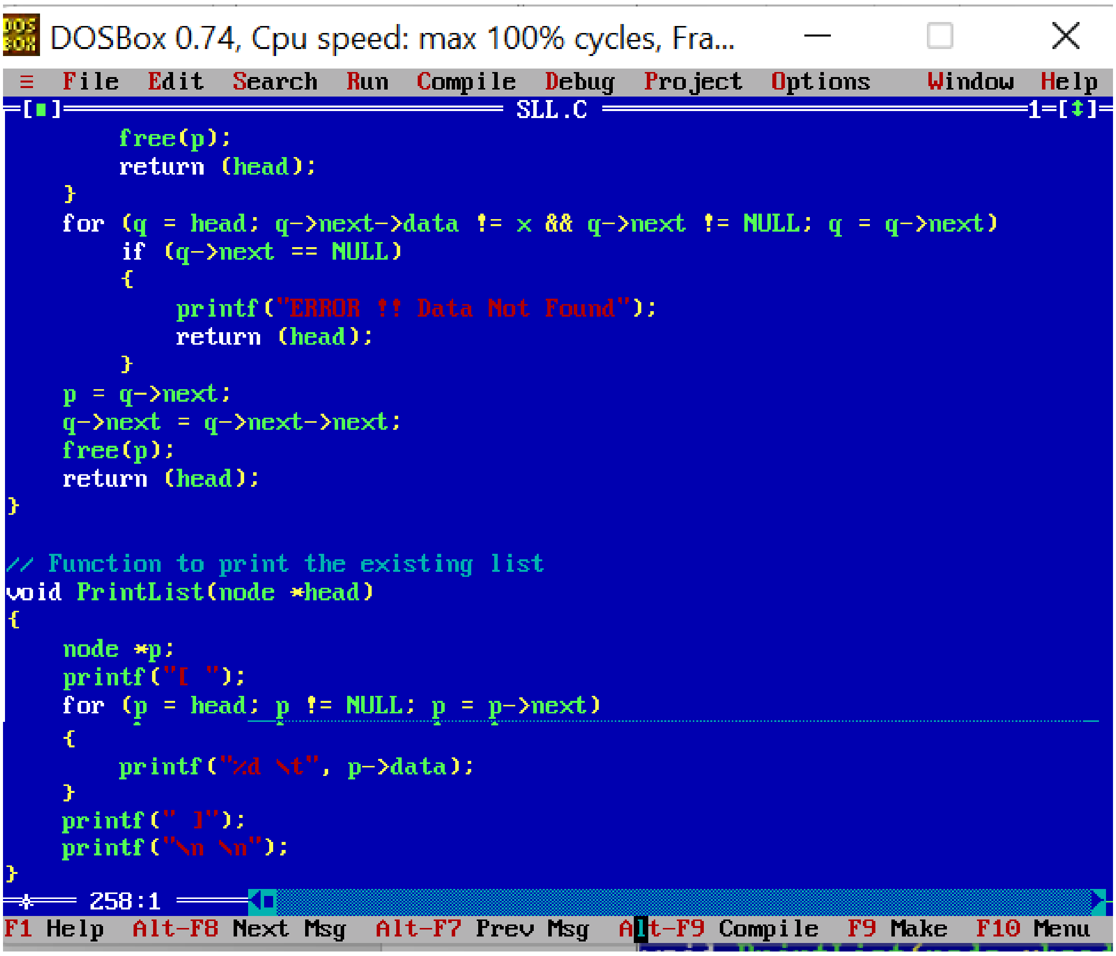
```
    scanf("%d", &y);
    for (q = head; q != NULL && q->data != y; q = q->next)
        ;
    if (q != NULL)
    {
        p->next = q->next;
        q->next = p;
    }
    else
        printf("ERROR !! Data Not Found");
    return (head);
}

// Function to delete element
node *Delete_beg(node *head)
{
    node *p, *q;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] SLL.C 1=[↕]
210:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
    }
    p = head;
    head = head->next;
    free(p);
    return (head);
}
node *Delete_end(node *head)
{
    node *p, *q;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    p = head;
    if (head->next == NULL)
    {
        head = NULL;
        free(p);
        return (head);
    }
```


DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] SLL.C 1=[↕]
231:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
    for (q = head; q->next->next != NULL; q = q->next)
        p = q->next;
    q->next = NULL;
    free(p);
    return (head);
}
node *Delete_mid(node *head)
{
    node *p, *q;
    int x, i;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    printf("Enter the data to be deleted: ");
    scanf("%d", &x);
    if (head->data == x)
    {
        p = head;
        head = head->next;
```

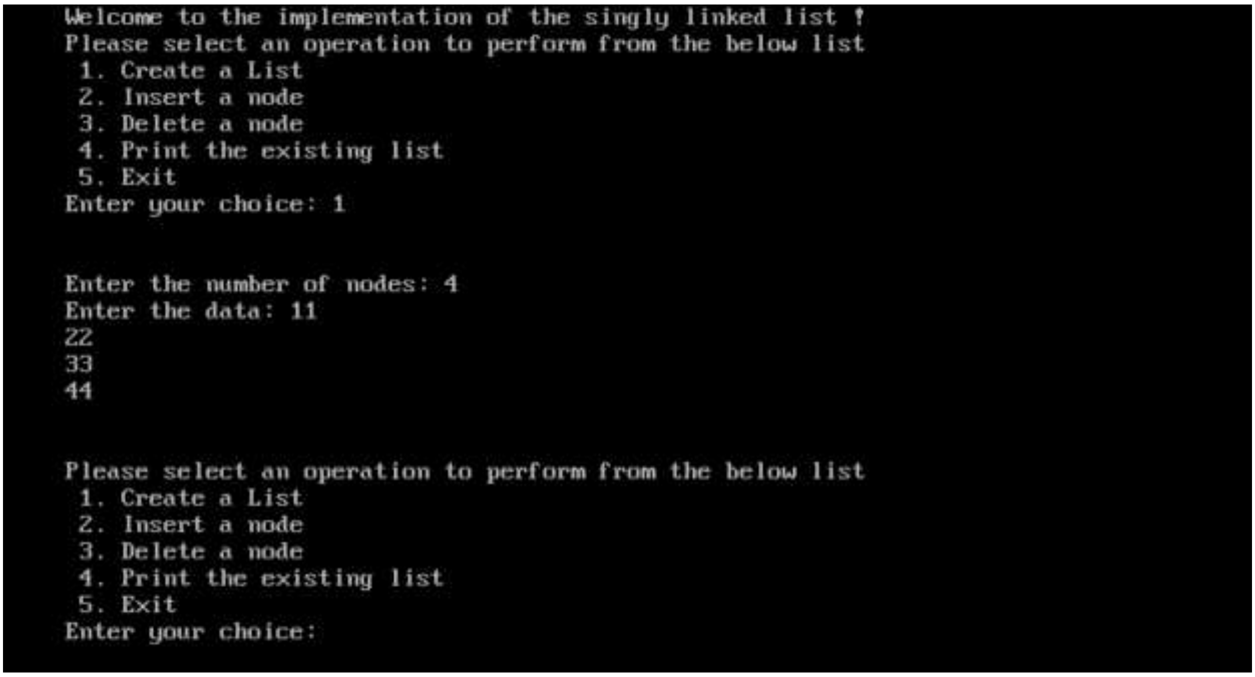

DOSBox 0.74, Cpu speed: max 100% cycles, Fra...
≡ File Edit Search Run Compile Debug Project Options Window Help
[■] SLL.C 1=[↕]
258:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
        free(p);
        return (head);
    }
    for (q = head; q->next->data != x && q->next != NULL; q = q->next)
        if (q->next == NULL)
        {
            printf("ERROR !! Data Not Found");
            return (head);
        }
    p = q->next;
    q->next = q->next->next;
    free(p);
    return (head);
}

// Function to print the existing list
void PrintList(node *head)
{
    node *p;
    printf("[ ");
    for (p = head; p != NULL; p = p->next)
    {
        printf("%d \t", p->data);
    }
    printf(" ]");
    printf("\n \n");
}
```

OUTPUT:-

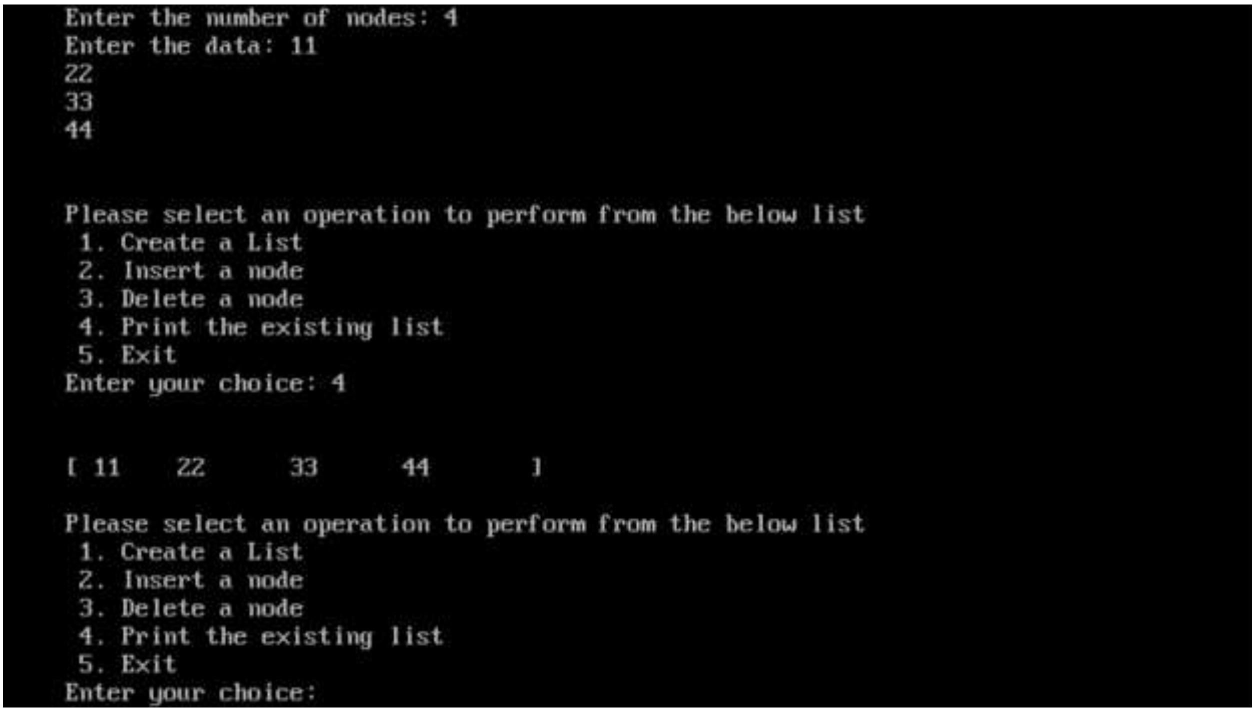
- SINGLY LINKED LIST:-

*Insertion in list:- (inserting 4 nodes 11,22,33,44)



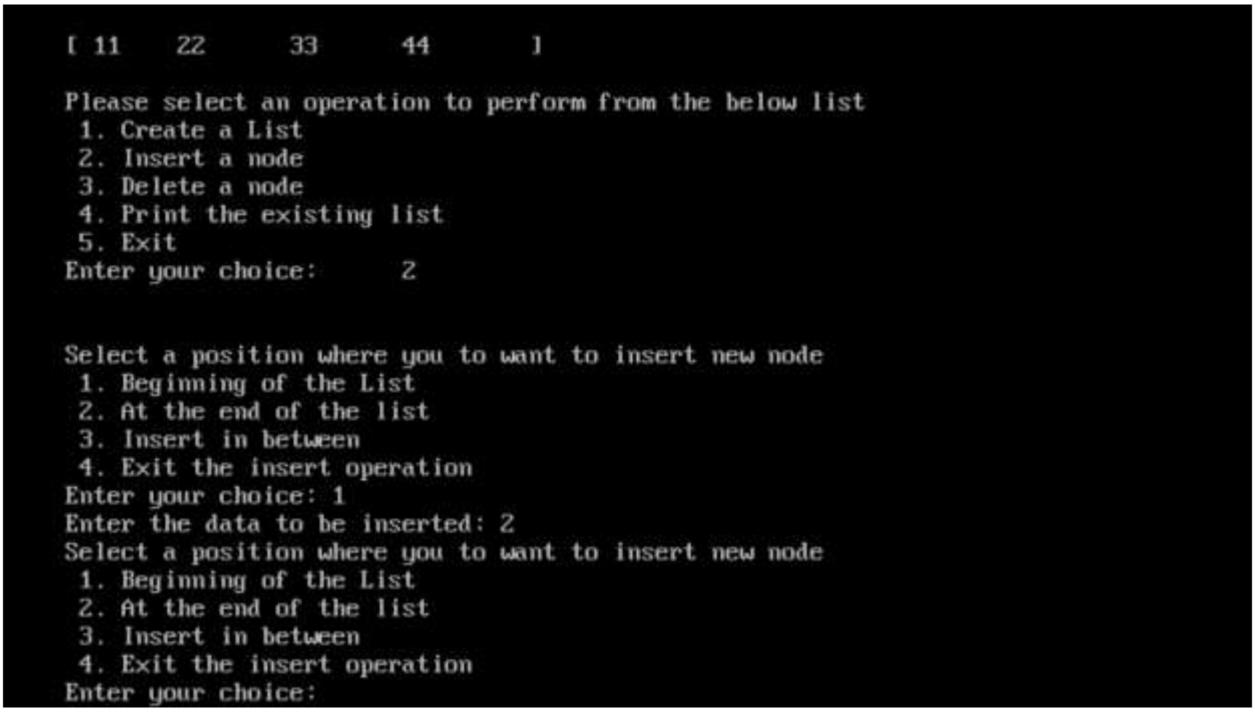
```
Welcome to the implementation of the singly linked list !
Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: 1


Enter the number of nodes: 4
Enter the data: 11
22
33
44


Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice:
```

*Displaying the list:-



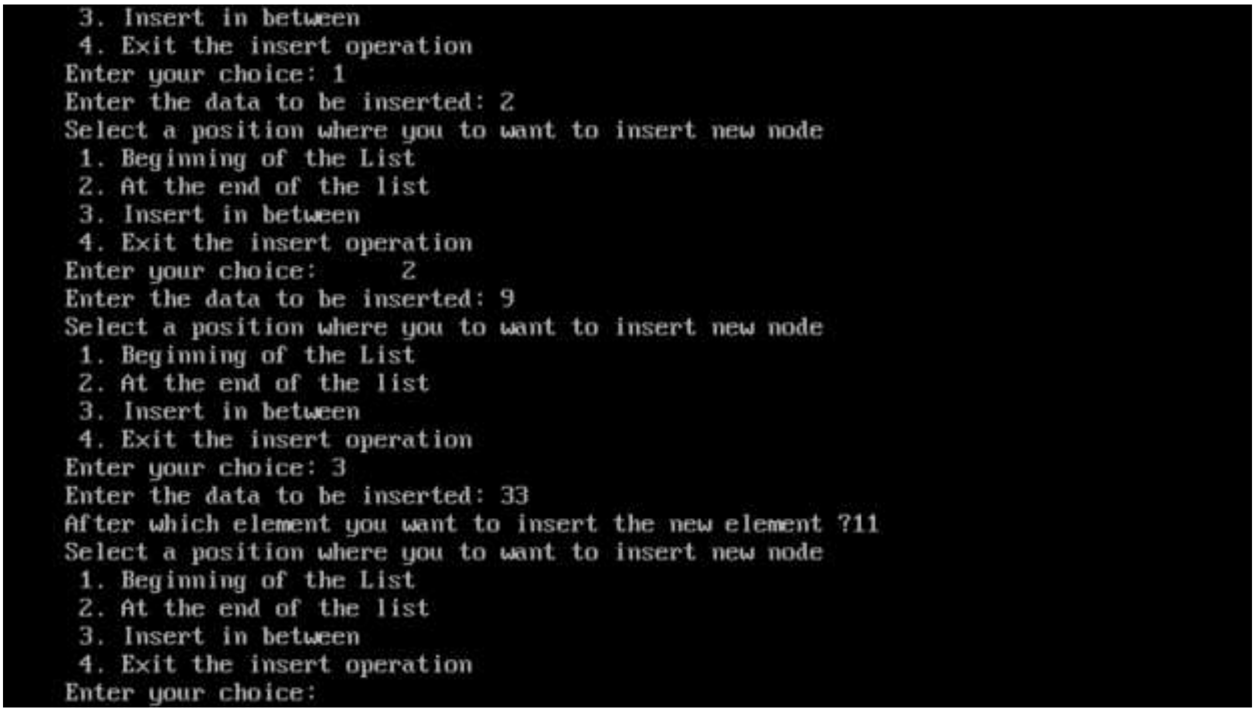
```
Enter the number of nodes: 4
Enter the data: 11
22
33
44


Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: 4


[ 11    22      33      44       ]

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice:
```

*Insertion:-



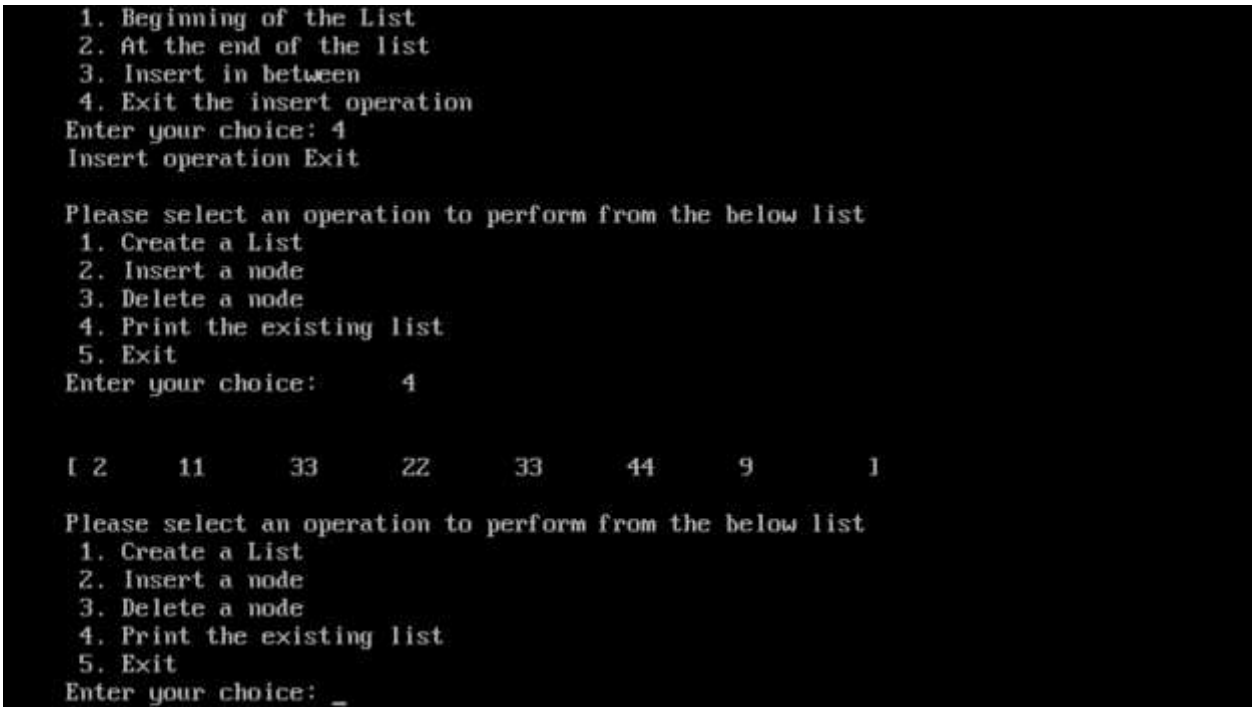
```
[ 11    22      33      44       ]

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice:      2


Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 1
Enter the data to be inserted: 2
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice:
```

*Inserting nodes at beginning, in the end and in between of the list:-



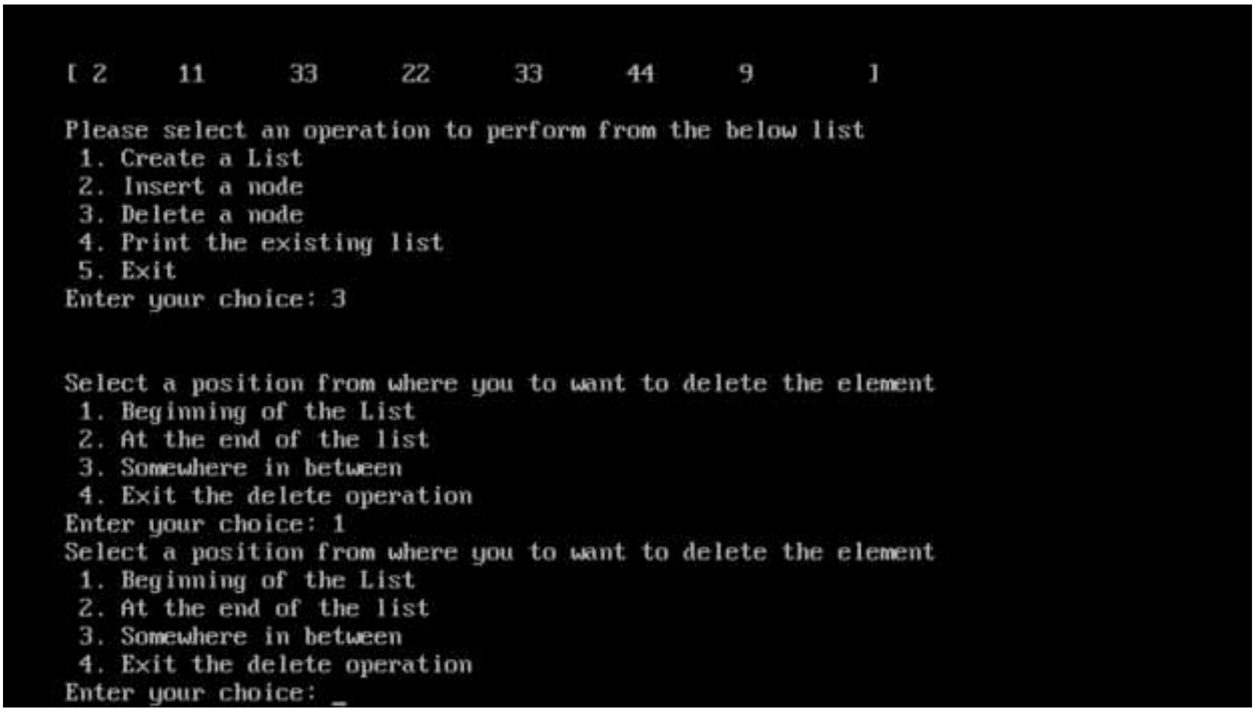
```
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 1
Enter the data to be inserted: 2
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice:      2
Enter the data to be inserted: 9
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 3
Enter the data to be inserted: 33
After which element you want to insert the new element ?11
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice:
```

*Displaying the list after insertion:-



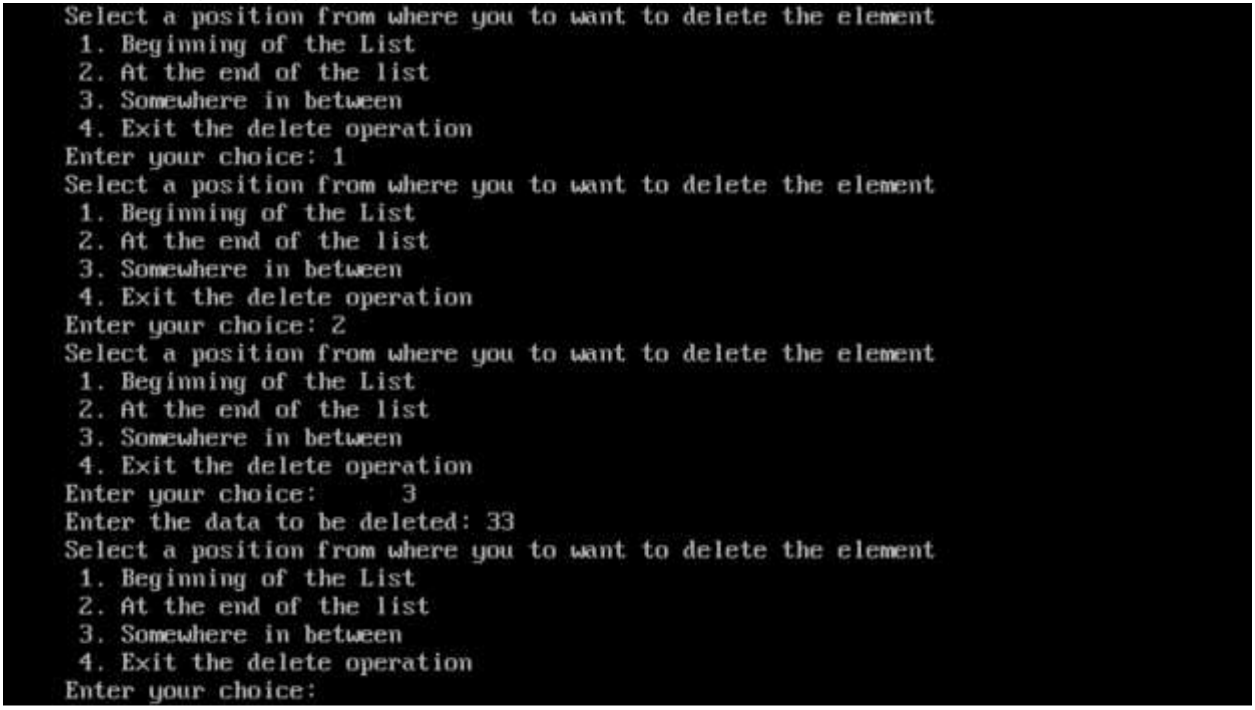
```
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 4
Insert operation Exit

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice:      4


[ 2     11      33      22      33      44      9       ]

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: _
```

*Deletion of node:-



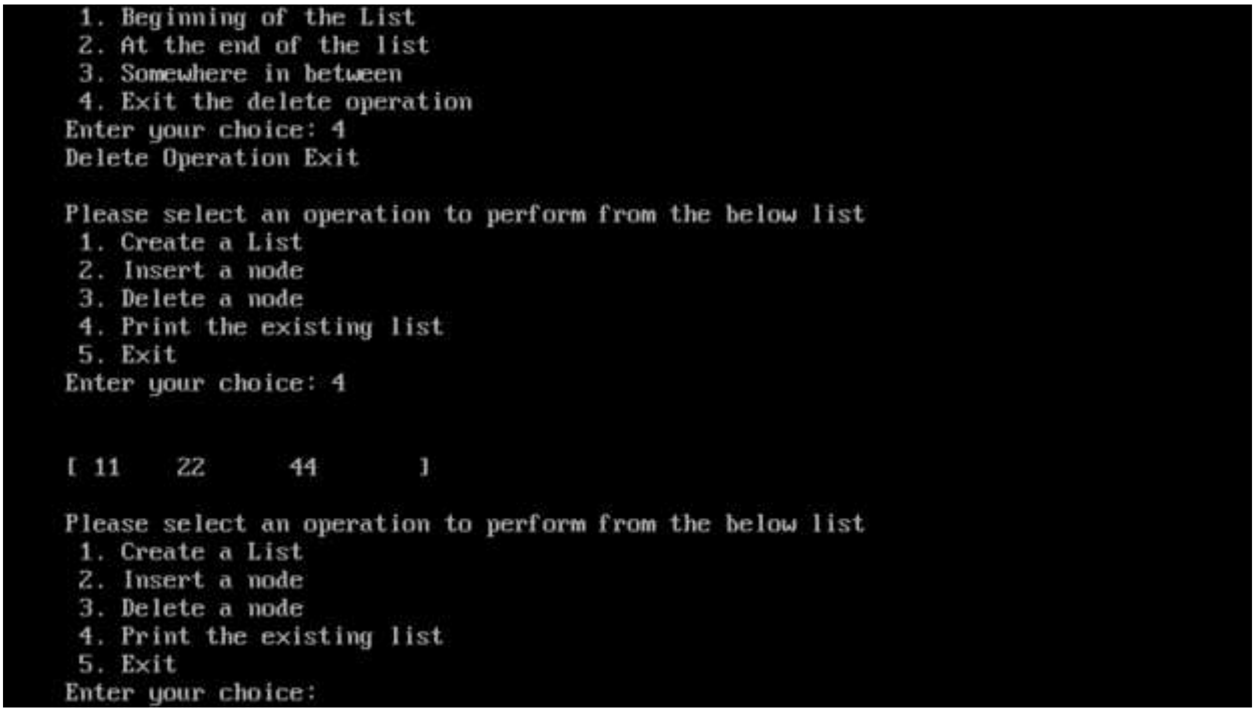
```
[ 2     11      33      22      33      44      9       ]

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: 3


Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. Somewhere in between
 4. Exit the delete operation
Enter your choice: 1
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. Somewhere in between
 4. Exit the delete operation
Enter your choice: _
```

*Deletion beginning ,from end and from in between of the list:-



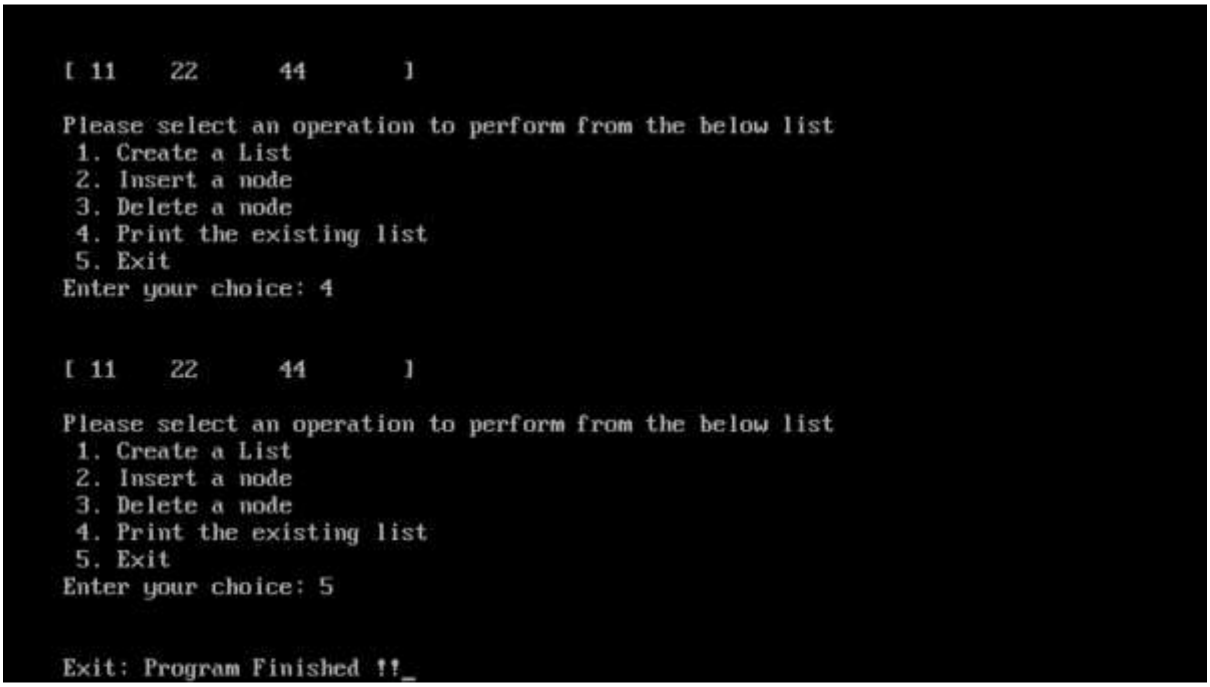
```
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. Somewhere in between
 4. Exit the delete operation
Enter your choice: 1
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. Somewhere in between
 4. Exit the delete operation
Enter your choice: 2
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. Somewhere in between
 4. Exit the delete operation
Enter your choice:      3
Enter the data to be deleted: 33
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. Somewhere in between
 4. Exit the delete operation
Enter your choice:
```

*Displaying list after deletion:-



```
 1. Beginning of the List
 2. At the end of the list
 3. Somewhere in between
 4. Exit the delete operation
Enter your choice: 4
Delete Operation Exit

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: 4


[ 11    22      44       ]

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice:
```

*Displaying the remaining node n Exit the program:-



```
[ 11    22      44       ]

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: 4


[ 11    22      44       ]

Please select an operation to perform from the below list
 1. Create a List
 2. Insert a node
 3. Delete a node
 4. Print the existing list
 5. Exit
Enter your choice: 5


Exit: Program Finished !!_
```