S.No: 2	Exp. Name: ***C program to which performs all operations on singly linked list.***	Date:2025-08-12

Aim:

Write a **C** program that uses functions to perform the following **operations on singly linked list.**

1. Insert at Begin.
2. Insert at End.
3. Insert node after the given node.
4. Delete first node.
5. Delete last node.
6. Delete node after the given node.
7. Delete at position.
8. Traversal.

Source Code:

SingleLL.c

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

#include "AllOperations.c"

void main() {
    NODE first = NULL;
    struct node * head = NULL;
    int x, pos, givenData, op;
    struct node * temp; // Declare temp variable here
    while (1) {
        printf("1.Insert At Begin\n2.Insert At End\n3.Insert node after the given nod
e\n");
        printf("4.Delete first node\n5.Delete last node\n6.Delete node after the give
n node\n7.Delete at position\n");
        printf("8.Traverse the List\n9.Exit\n");
        printf("Enter your option: ");
        scanf("%d", & op);
        switch (op) {
        case 1:
            printf("Enter an element: ");
            scanf("%d", & x);
            first = insertAtBegin(first, x);
            break;
        case 2:
            printf("Enter an element: ");
            scanf("%d", & x);
            first = insertAtEnd(first, x);
            break;
        case 3:
            printf("Enter the data of the given node: ");
            scanf("%d", & givenData);
            printf("Enter the data to be inserted: ");
            scanf("%d", & x);
            temp = first; // Reuse temp variable here
            while (temp != NULL && temp -> data != givenData) {
                temp = temp -> next;
```

```
            }
            if (temp == NULL) {
                printf("Given node not found in the list.\n");
            } else {
                insertAfterNode(temp, x);
            }
            break;
        case 4:
            if (first == NULL) {
                printf("Single Linked List is empty so deletion is not possible\n");
            } else {
                first = deleteAtBegin(first);
            }
            break;
        case 5:
            if (first == NULL) {
                printf("Single Linked List is empty so deletion is not possible\n");
            } else {
                first = deleteAtEnd(first);
            }
            break;
        case 6:
            printf("Enter the data of the given node: ");
            scanf("%d", & givenData);
            temp = first; // Reuse temp variable here
            while (temp != NULL && temp -> data != givenData) {
                temp = temp -> next;
            }
            if (temp == NULL) {
                printf("Given node not found in the list.\n");
            } else {
                deleteAfterNode(temp);
            }
            break;
        case 7:
            if (first == NULL) {
                printf("Single Linked List is empty so deletion is not possible\n");
            } else {
                printf("Enter position: ");
                scanf("%d", & pos);
                first = deleteAtPosition(first, pos);
            }
            break;
        case 8:
            if (first == NULL) {
                printf("Single Linked List is empty\n");
            } else {
                printf("The elements in SLL are: ");
                traverseList(first);
            }
            break;
        case 9:
            exit(0);
        }
    }
}
```

AllOperations.c

```
struct node {
    int data;
    struct node * next;
};
typedef struct node * NODE;

NODE createNode() {
    NODE temp;
    temp = (NODE) malloc(sizeof(struct node));
    temp -> next = NULL;
    return temp;
}

NODE insertAtBegin(NODE first, int x) {
   NODE newNode = createNode();
   newNode->data = x;
   newNode->next = first;
   first = newNode;
   return newNode;

}

NODE insertAtEnd(NODE first, int x) {
   NODE newNode = createNode();
   newNode->data = x;

   if(first == NULL)









      return newNode;
   NODE temp = first;
   while(temp->next != NULL){
      temp = temp->next;
   }
   temp->next = newNode;
   return first;
}

void insertAfterNode(struct node * givenNode, int data) {
   if(givenNode == NULL){
      printf("Given node not found in the list.\n");
      return;
   }
```


ID: 24CS079
2024-2028-CSE-B
KPR Institute of Engineering and Technology

```
   NODE newNode = createNode();
   newNode->data = data;
   newNode->next = givenNode->next;
   givenNode->next = newNode;

   printf("Inserted node with data %d after the given node.\n", data);
}

NODE deleteAtBegin(NODE first) {
   if(first == NULL){
      printf("List is empty\n");
      return NULL;
   }
   NODE temp = first;
   first = first->next;
   printf("The deleted element from SLL : %d\n", temp->data);
   free(temp);
   return first;

}

NODE deleteAtEnd(NODE first) {
   if(first == NULL){
      printf("List is empty\n");
      return first;
   }
   if(first->next == NULL){
      printf("The deleted item from SLL : %d\n", first->data);
      free(first);
      return NULL;
   }
   NODE temp = first;
   while(temp->next->next!=NULL){
      temp = temp->next;
   }
   int dt = temp->next->data;
   free(temp->next);
   temp->next=NULL;
   printf("The deleted item from SLL : %d\n", dt);
   return first;

}

void deleteAfterNode(struct node * givenNode) {
   if(givenNode == NULL || givenNode->next == NULL){
      printf("Cannot delete. Given node is NULL or the next node is NULL.\n");
      return;
   }
   NODE temp = givenNode->next;
   int dt = temp->data;
   givenNode->next = temp->next;
   printf("Deleted node with data: %d\n", dt);
   free(temp);

}
```

```
NODE deleteAtPosition(NODE first, int pos) {
   if(first == NULL){
      printf("List is empty\n");
      return NULL;
   }
   if(pos<0){
      printf("Invalid position\n");
   }
   if(pos == 1){
      return deleteAtBegin(first);
   }

   NODE temp = first;
   for(int i = 1; temp!=NULL && i<pos-1; i++){
      temp=temp->next;
   }
   if(temp == NULL || temp->next == NULL){
      printf("No such position in SLL so deletion is not possible\n");
      return first;
   }
   NODE td = temp->next;
   int dv = td->data;
   temp->next = td->next;
   free(td);
   printf("The deleted element from SLL : %d\n", dv);
   return first;
}

void traverseList(NODE first) {
    NODE temp = first;
    while (temp != NULL) {
        printf("%d --> ", temp -> data);
        temp = temp -> next;
    }
    printf("NULL\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1.Insert At Begin 3
2.Insert At End 3
3.Insert node after the given node 3
4.Delete first node 3
5.Delete last node 3
6.Delete node after the given node 3
7.Delete at position 3
8.Traverse the List 3
9.Exit 3
Enter your option: 3
Enter the data of the given node: 15
Enter the data to be inserted: 25

```
Given node not found in the list. 6
1.Insert At Begin 6
2.Insert At End 6
3.Insert node after the given node 6
4.Delete first node 6
5.Delete last node 6
6.Delete node after the given node 6
7.Delete at position 6
8.Traverse the List 6
9.Exit 6
Enter your option:  6
Enter the data of the given node:  48
Given node not found in the list. 1
1.Insert At Begin 1
2.Insert At End 1
3.Insert node after the given node 1
4.Delete first node 1
5.Delete last node 1
6.Delete node after the given node 1
7.Delete at position 1
8.Traverse the List 1
9.Exit 1
Enter your option:  1
Enter an element:  101
1.Insert At Begin 1
2.Insert At End 1
3.Insert node after the given node 1
4.Delete first node 1
5.Delete last node 1
6.Delete node after the given node 1
7.Delete at position 1
8.Traverse the List 1
9.Exit 1
Enter your option:  1
Enter an element:  102
1.Insert At Begin 1
2.Insert At End 1
3.Insert node after the given node 1
4.Delete first node 1
5.Delete last node 1
6.Delete node after the given node 1
7.Delete at position 1
8.Traverse the List 1
9.Exit 1
Enter your option:  1
Enter an element:  103
1.Insert At Begin 8
2.Insert At End 8
3.Insert node after the given node 8
4.Delete first node 8
5.Delete last node 8
```

```
6.Delete node after the given node 8
7.Delete at position 8
8.Traverse the List 8
9.Exit 8
Enter your option:  8
The elements in SLL are: 103 --> 102 --> 101 --> NULL 2
1.Insert At Begin 2
2.Insert At End 2
3.Insert node after the given node 2
4.Delete first node 2
5.Delete last node 2
6.Delete node after the given node 2
7.Delete at position 2
8.Traverse the List 2
9.Exit 2
Enter your option:  2
Enter an element:  99
1.Insert At Begin 2
2.Insert At End 2
3.Insert node after the given node 2
4.Delete first node 2
5.Delete last node 2
6.Delete node after the given node 2
7.Delete at position 2
8.Traverse the List 2
9.Exit 2
Enter your option:  2
Enter an element:  98
1.Insert At Begin 2
2.Insert At End 2
3.Insert node after the given node 2
4.Delete first node 2
5.Delete last node 2
6.Delete node after the given node 2
7.Delete at position 2
8.Traverse the List 2
9.Exit 2
Enter your option:  2
Enter an element:  97
1.Insert At Begin 8
2.Insert At End 8
3.Insert node after the given node 8
4.Delete first node 8
5.Delete last node 8
6.Delete node after the given node 8
7.Delete at position 8
8.Traverse the List 8
9.Exit 8
Enter your option:  8
The elements in SLL are: 103 --> 102 --> 101 --> 99 --> 98 --> 97 --> NULL 3
1.Insert At Begin 3
```

2.Insert At End 3
3.Insert node after the given node 3
4.Delete first node 3
5.Delete last node 3
6.Delete node after the given node 3
7.Delete at position 3
8.Traverse the List 3
9.Exit 3
Enter your option: 3
Enter the data of the given node: 97
Enter the data to be inserted: 96
Inserted node with data 96 after the given node. 8
1.Insert At Begin 8
2.Insert At End 8
3.Insert node after the given node 8
4.Delete first node 8
5.Delete last node 8
6.Delete node after the given node 8
7.Delete at position 8
8.Traverse the List 8
9.Exit 8
Enter your option: 8
The elements in SLL are: 103 --> 102 --> 101 --> 99 --> 98 --> 97 --> 96 --> NULL 4
1.Insert At Begin 4
2.Insert At End 4
3.Insert node after the given node 4
4.Delete first node 4
5.Delete last node 4
6.Delete node after the given node 4
7.Delete at position 4
8.Traverse the List 4
9.Exit 4
Enter your option: 4
The deleted element from SLL : 103 5
1.Insert At Begin 5
2.Insert At End 5
3.Insert node after the given node 5
4.Delete first node 5
5.Delete last node 5
6.Delete node after the given node 5
7.Delete at position 5
8.Traverse the List 5
9.Exit 5
Enter your option: 5
The deleted item from SLL : 96 8
1.Insert At Begin 8
2.Insert At End 8
3.Insert node after the given node 8
4.Delete first node 8
5.Delete last node 8
6.Delete node after the given node 8

7.Delete at position 8
8.Traverse the List 8
9.Exit 8
Enter your option: 8
The elements in SLL are: 102 --> 101 --> 99 --> 98 --> 97 --> NULL 6
1.Insert At Begin 6
2.Insert At End 6
3.Insert node after the given node 6
4.Delete first node 6
5.Delete last node 6
6.Delete node after the given node 6
7.Delete at position 6
8.Traverse the List 6
9.Exit 6
Enter your option: 6
Enter the data of the given node: 97
Cannot delete. Given node is NULL or the next node is NULL. 6
1.Insert At Begin 6
2.Insert At End 6
3.Insert node after the given node 6
4.Delete first node 6
5.Delete last node 6
6.Delete node after the given node 6
7.Delete at position 6
8.Traverse the List 6
9.Exit 6
Enter your option: 6
Enter the data of the given node: 96
Given node not found in the list. 6
1.Insert At Begin 6
2.Insert At End 6
3.Insert node after the given node 6
4.Delete first node 6
5.Delete last node 6
6.Delete node after the given node 6
7.Delete at position 6
8.Traverse the List 6
9.Exit 6
Enter your option: 6
Enter the data of the given node: 101
Deleted node with data: 99 8
1.Insert At Begin 8
2.Insert At End 8
3.Insert node after the given node 8
4.Delete first node 8
5.Delete last node 8
6.Delete node after the given node 8
7.Delete at position 8
8.Traverse the List 8
9.Exit 8
Enter your option: 8

```
The elements in SLL are: 102 --> 101 --> 98 --> 97 --> NULL 7
1.Insert At Begin 7
2.Insert At End 7
3.Insert node after the given node 7
4.Delete first node 7
5.Delete last node 7
6.Delete node after the given node 7
7.Delete at position 7
8.Traverse the List 7
9.Exit 7
Enter your option:  7
Enter position:  5
No such position in SLL so deletion is not possible 7
1.Insert At Begin 7
2.Insert At End 7
3.Insert node after the given node 7
4.Delete first node 7
5.Delete last node 7
6.Delete node after the given node 7
7.Delete at position 7
8.Traverse the List 7
9.Exit 7
Enter your option:  7
Enter position:  6
No such position in SLL so deletion is not possible 7
1.Insert At Begin 7
2.Insert At End 7
3.Insert node after the given node 7
4.Delete first node 7
5.Delete last node 7
6.Delete node after the given node 7
7.Delete at position 7
8.Traverse the List 7
9.Exit 7
Enter your option:  7
Enter position:  4
The deleted element from SLL : 97 8
1.Insert At Begin 8
2.Insert At End 8
3.Insert node after the given node 8
4.Delete first node 8
5.Delete last node 8
6.Delete node after the given node 8
7.Delete at position 8
8.Traverse the List 8
9.Exit 8
Enter your option:  8
The elements in SLL are: 102 --> 101 --> 98 --> NULL 9
1.Insert At Begin 9
2.Insert At End 9
3.Insert node after the given node 9
```

4.Delete first node 9
5.Delete last node 9
6.Delete node after the given node 9
7.Delete at position 9
8.Traverse the List 9
9.Exit 9
Enter your option: 9