DELETION IN SINGLY LINKED LIST: -

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

struct node {
int data;
struct node *next;
};

struct node *head = NULL;

void begin_delete();
void last_delete();
void random_delete();
void display();

int main() {
int choice = 0;

// Creating linked list manually
struct node *first, *second, *third, *fourth;
first = (struct node*)malloc(sizeof(struct node));
second = (struct node*)malloc(sizeof(struct node));
third = (struct node*)malloc(sizeof(struct node));
fourth = (struct node*)malloc(sizeof(struct node));

first->data = 10;
first->next = second;
second->data = 20;
second->next = third;
third->data = 30;
third->next = fourth;
fourth->data = 40;
fourth->next = NULL;
head = first;

printf("\nInitial Linked List:\n");
display();
```

```
38
39  while (choice != 4) {
40  printf("\n********* Main Menu *********\n");
41  printf("1. Delete from Beginning\n");
42  printf("2. Delete from End\n");
43  printf("3. Delete from Specific Position\n");
44  printf("4. Exit\n");
45  printf("\nEnter your choice: ");
46  scanf("%d", &choice);
47
48  switch (choice) {
49  case 1:
50  begin_delete();
51  display();
52  break;
53  case 2:
54  last_delete();
55  display();
56  break;
57  case 3:
58  random_delete();
59  display();
60  break;
61  case 4:
62  printf("\nExiting...\n");
63  break;
64  default:
65  printf("\nPlease enter a valid choice (1-4)!\n");
66  }
67  }
68  return 0;
69  }
70
71  void begin_delete() {
72  struct node *ptr;
73  if (head == NULL) {
```

```
74      printf("\nList is empty\n");
75      } else {
76      ptr = head;
77      head = ptr->next;
78      free(ptr);
79      printf("\nNode deleted from the beginning\n");
80      }
81      }
82
83      void last_delete() {
84      struct node *ptr, *ptr1;
85      if (head == NULL) {
86      printf("\nList is empty\n");
87      } else if (head->next == NULL) {
88      free(head);
89      head = NULL;
90      printf("\nOnly node of the list deleted\n");
91      } else {
92      ptr = head;
93      while (ptr->next->next != NULL) {
94      ptr = ptr->next;
95      }
96      ptr1 = ptr->next;
97      free(ptr1);
98      ptr->next = NULL;
99      printf("\nDeleted node from the last\n");
100     }
101     }
102
103     void random_delete() {
104     struct node *ptr, *ptr1;
105     int loc, i;
106     if (head == NULL) {
107     printf("\nList is empty\n");
108     return;
109     }
```

```
110     printf("Enter the location of the node after which you want to perform deletion: ");
111     scanf("%d", &loc);
112     ptr = head;
113     for (i = 1; i < loc; i++) {
114     if (ptr == NULL || ptr->next == NULL) {
115     printf("\nCan't delete\n");
116     return;
117     }
118     ptr = ptr->next;
119     }
120     ptr1 = ptr->next;
121     ptr->next = ptr1->next;
122     free(ptr1);
123     printf("\nDeleted node at location %d\n", loc + 1);
124     }
125
126     void display() {
127     struct node *ptr;
128     ptr = head;
129     if (ptr == NULL) {
130     printf("\nNothing to print\n");
131     } else {
132     printf("\nPrinting values...\n");
133     while (ptr != NULL) {
134     printf("%d ", ptr->data);
135     ptr = ptr->next;
136     }
137     printf("\n");
138     }
139     }
```

OUTPUT: -

```
Initial Linked List:

Printing values...
10 20 30 40

********* Main Menu *********
1. Delete from Beginning
2. Delete from End
3. Delete from Specific Position
4. Exit

Enter your choice: 1

Node deleted from the beginning

Printing values...
20 30 40
```