

```

1: #include<stdio.h>
2: #include<stdlib.h>
3: struct Node;
4: typedef struct Node * PtrToNode;
5: typedef PtrToNode List;
6: typedef PtrToNode Position;
7: struct Node
8: {
9:     int e;
10:    Position next;
11: };
12: void Insert(int x, List l, Position p)
13: {
14:    Position TmpCell;
15:    TmpCell = (struct Node*) malloc(sizeof(struct Node));
16:    if(TmpCell == NULL)
17:        printf("Memory out of space\n");
18:    else
19:    {
20:        TmpCell->e = x;
21:        TmpCell->next = p->next;
22:        p->next = TmpCell;
23:    }
24: }
25: int isLast(Position p)
26: {
27:    return (p->next == NULL);
28: }
29: Position FindPrevious(int x, List l)
30: {
31:    Position p = l;
32:    while(p->next != NULL && p->next->e != x)
33:        p = p->next;
34:    return p;
35: }
36: void Delete(int x, List l)
37: {
38:    Position p, TmpCell;
39:    p = FindPrevious(x, l);
40:    if(!isLast(p))
41:    {
42:        TmpCell = p->next;
43:        p->next = TmpCell->next;
44:        free(TmpCell);
45:    }
46:    else
47:        printf("Element does not exist!!!\n");
48: }
49: void Display(List l)
50: {
51:    printf("The list element are :: ");
52:    Position p = l->next;
53:    while(p != NULL)
54:    {
55:        printf("%d -> ", p->e);
56:        p = p->next;
57:    }
58: }
59: void Merge(List l, List l1)
60: {
61:    int i, n, x, j;

```

```

62:     Position p;
63:     printf("Enter the number of elements to be merged :: ");
64:     scanf("%d",&n);
65:     for(i = 1; i <= n; i++)
66:     {
67:         p = l1;
68:         scanf("%d", &x);
69:         for(j = 1; j < i; j++)
70:             p = p->next;
71:         Insert(x, l1, p);
72:     }
73:     printf("The new List :: ");
74:     Display(l1);
75:     printf("The merged List ::");
76:     p = l;
77:     while(p->next != NULL)
78:     {
79:         p = p->next;
80:     }
81:     p->next = l1->next;
82:     Display(l);
83: }
84: int main()
85: {
86:     int x, pos, ch, i;
87:     List l, l1;
88:     l = (struct Node *) malloc(sizeof(struct Node));
89:     l->next = NULL;
90:     List p = l;
91:     printf("\n\nLINKED LIST IMPLEMENTATION OF LIST ADT\n\n");
92:     do
93:     {
94:         printf("\n\n1. INSERT\t2. DELETE\t3. MERGE\t4. PRINT\t5. QUIT\n\nEnter the choice :: ");
95:         scanf("%d", &ch);
96:         switch(ch)
97:         {
98:             case 1:
99:                 p = l;
100:                 printf("Enter the element to be inserted :: ");
101:                 scanf("%d",&x);
102:                 printf("Enter the position of the element :: ");
103:                 scanf("%d",&pos);
104:                 for(i = 1; i < pos; i++)
105:                 {
106:                     p = p->next;
107:                 }
108:                 Insert(x,l,p);
109:                 break;
110:             case 2:
111:                 p = l;
112:                 printf("Enter the element to be deleted :: ");
113:                 scanf("%d",&x);
114:                 Delete(x,p);
115:                 break;
116:             case 3:
117:                 l1 = (struct Node *) malloc(sizeof(struct Node));
118:                 l1->next = NULL;
119:                 Merge(l, l1);
120:                 break;
121:             case 4:
122:                 Display(l);

```

```
123:         break;
124:     }
125: }
126: while(ch<5);
127: return 0;
128: }
```