

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  typedef struct node
4  {
5      int data;
6      struct node *next;
7  }node;
8  node *head;
9  void create();
10 void display();
11 void Insert_At_Pos(int data,int pos);
12 void Delete_All_Nodes();
13 void reverse();
14
15 int main()
16 {
17     int choice;
18     while(1)
19     {
20         printf("\nEnter choice:\n1.Create SLL\n2.Display\n3.Insert at any
position\n4.Delete all nodes\n5.Reverse the linked list\n6.Exit\n");
21         scanf("%d",&choice);
22         int pos,data;
23         switch(choice)
24         {
25             case 1: create();
26                     break;
27             case 2: display();
28                     break;
29             case 3: printf("\nEnter Position where you want to enter: ");
30                     scanf("%d",&pos);
31                     printf("\nEnter Data: ");
32                     scanf("%d",&data);
33                     Insert_At_Pos(data,pos);
34                     break;
35             case 4: Delete_All_Nodes();
36                     break;
37             case 5: reverse();
38                     break;
39             case 6: return 0;
40             default: printf("\nInvalid Input\n");
41                     return 0;
42         }
43     }
44 }
45
46
47
48 void create()
49 {
50     int num,i;
51     node *new_node,*ptr;
52     printf("\nEnter Number of data you want to enter: ");
53     scanf("%d",&num);
54     head=ptr=NULL;
55     while(num)
56     {
57         new_node=(node *)malloc(sizeof(node));
58         printf("\nEnter data: ");
59         scanf("%d",&new_node->data);
60         new_node->next=NULL;
61         if(head==NULL)
62             head=ptr=new_node;
63         else
64             {
65                 ptr->next=new_node;

```

```

66         ptr=ptr->next;
67     }
68     num--;
69 }
70 printf("\nLinked List Created\n");
71 printf(
"\n\n";
72 }
73
74 void display()
75 {
76     node *ptr;
77     ptr=head;
78     if(head==NULL)
79     {
80         printf("\nThe list is empty\n");
81         return ;
82     }
83     printf("\nDisplaying Elements of Linked List: ");
84     while(ptr)
85     {
86         printf("%d ",ptr->data);
87         ptr=ptr->next;
88     }
89     printf(
"\n\n";
90 }
91
92 void Delete_All_Nodes()
93 {
94     node *ptr,*preptr;
95     ptr=head;
96     preptr=NULL;
97     if(ptr==NULL)
98     {
99         printf("\nThe list is Empty\n");
100        return;
101    }
102    do
103    {
104        preptr=ptr;
105        ptr=ptr->next;
106        head=ptr;
107        free(preptr);
108    }while(ptr!=NULL);
109
110    printf("\nAll nodes are deleted\n");
111    printf(
"\n\n";
112 }
113
114 void Insert_At_Pos(int info,int pos)
115 {
116     node *ptr,*new_node;
117     ptr=head;
118     new_node=(node *)malloc(sizeof(node));
119     if(new_node==NULL)
120     {
121         printf("Overflow Condition");
122         return;
123     }
124     int i;
125     new_node->data=info;
126     new_node->next=NULL;
127     if(pos==1)
128     {

```

```

129         new_node->next=head;
130         head=new_node;
131     }
132     else if(pos>0)
133     {
134         for(i=1;i<pos-1 && ptr!=NULL ;i++)
135             ptr=ptr->next;
136         if(ptr)
137         {
138             new_node->next=ptr->next;
139             ptr->next=new_node;
140         }
141     }
142     else
143         printf("\nInvalid Position\n");
144
145     printf(
146         "\n\n");
147
148     void reverse()
149     {
150         node *p,*q,*r;           //p : preptr , q : ptr , r : postptr
151         p=q=NULL;
152         r=head;
153         while(r!=NULL)
154         {
155             p=q;
156             q=r;
157             r=r->next;
158             q->next=p;
159         }
160         head=q;
161
162         printf("\nLinked list reversed\n");
163         printf(
164             "\n\n");

```