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

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

typedef struct node *NODE;

NODE create_node();
NODE insert_end(NODE head);
NODE insert_front(NODE head);
NODE delete_end(NODE head);
NODE delete_front(NODE head);
int count_nodes(NODE head);
void search_node(NODE head);
NODE search_and_delete(NODE head);
void display_list(NODE head);
NODE insert_position(NODE head);
NODE delete_position(NODE head);

int main()
{
    NODE head=NULL;
    int ch, count;
    while(1)
    {
        printf("\nSingly linked list menu\n");
        printf("1: Insert End 2: Insert Front 3: Delete End 4: Delete front 5: Count Nodes 6: Search Node 7: Search and Delete 8: Display 9: Insert position 10: Delete Pos 11: Exit\n");
        printf("Enter your choice\n");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1: head = insert_end(head);
                    break;
            case 2: head = insert_front(head);
                    break;
            case 3: head = delete_end(head);
                    break;
            case 4: head = delete_front(head);
                    break;
            case 5: count  = count_nodes(head);
                    printf("Number of nodes is %d\n", count);
                    break;
            case 6: search_node(head);
                    break;
            case 7: head = search_and_delete(head);
                    break;
            case 8: display_list(head);
                    break;
            case 9: head = insert_position(head);
                    break;
            case 10: head = delete_position(head);
                    break;
            case 11: exit(0);
                    break;
            default: printf("Invalid choice\n");
                    break;
        }
    }
}

NODE create_node()
{
    NODE newnode = (NODE) malloc (sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Memory not allocated\n");
    }
    printf("Enter data\n");
    scanf("%d",&newnode->data);
    newnode->link= NULL;
    return newnode;
}
```

```
77
78  NODE insert_end(NODE head)
79  {
80      NODE cur, newnode;
81      newnode = create_node();
82      if(head==NULL)
83      {
84          head= newnode;
85      }
86      else
87      {
88          cur = head;
89          while(cur->link !=NULL)
90          {
91              cur = cur->link;
92          }
93          cur->link = newnode;
94      }
95      return head;
96  }
97
98  NODE insert_position(NODE head)
99  {
100     NODE prev, cur;
101     NODE newnode = create_node();
102     int pos;
103     int count = count_nodes(head);
104     printf("Enter position\n");
105     scanf("%d",&pos);
106     if(pos==1 && head == NULL)
107     {
108         head = newnode;
109     }
110     else if(pos==1)
111     {
112         newnode->link = head;
113         head = newnode;
114     }
115     else if(pos == count+1)
116     {
117         cur = head;
118         while(cur->link != NULL)
119         {
120             cur = cur->link;
121         }
122         cur->link = newnode;
123     }
124     else if(pos>1 && pos<=count)
125     {
126
127             prev=NULL;
128             cur=head;
129             for(int i =1; i<pos-1; i++)
130             {
131                 prev=cur;
132                 cur = cur->link;
133             }
134             prev->link = newnode;
135             newnode->link = cur;
136     }
137     else
138     {
139         printf("Invalid position\n");
140     }
141     return head;
142 }
143
144
145
146 NODE insert_front(NODE head)
147 {
148     NODE cur, newnode;
149     newnode = create_node();
150     if(head==NULL)
151     {
152         head= newnode;
153     }
```

```
154         else
155         {
156             newnode->link = head;
157             head = newnode;
158         }
159         return head;
160     }
161
162     NODE delete_end(NODE head)
163     {
164         NODE prev , cur;
165         if(head==NULL)
166         {
167             printf("Singly linked list es empty\n");
168         }
169         else if(head->link == NULL)
170         {
171             printf("Deleted %d\n", head->data);
172             free(head);
173             head=NULL;
174         }
175         else
176         {
177             prev=NULL;
178             cur=head;
179             while(cur->link != NULL)
180             {
181                 prev = cur;
182                 cur = cur->link;
183             }
184             prev->link = NULL;
185             printf("Deleted %d\n", cur->data);
186             free(cur);
187         }
188         return head;
189     }
190
191     NODE delete_front(NODE head)
192     {
193         NODE cur;
194         if(head==NULL)
195         {
196             printf("Singly linked list is empty\n");
197         }
198         else if(head->link == NULL)
199         {
200             printf("Deleted %d\n", head->data);
201             free(head);
202             head=NULL;
203         }
204         else
205         {
206             cur = head;
207             head = head->link;
208             printf("Deleted %d\n", cur->data);
209             cur->link = NULL;
210             free(cur);
211         }
212         return head;
213     }
214
215     int count_nodes(NODE head)
216     {
217         int count=0;
218         NODE cur;
219         if(head==NULL)
220         {
221             return count;
222         }
223         else
224         {
225             cur = head;
226             while(cur != NULL)
227             {
228                 count++;
229                 cur = cur-> link;
230             }
```

```
231         }
232         return count;
233     }
234
235     void display_list(NODE head)
236     {
237         NODE cur;
238         if(head==NULL)
239         {
240             printf("\nSingly linked list empty\n");
241         }
242         else
243         {
244             printf("\nSingly linked list is \n");
245             cur = head;
246             while(cur != NULL)
247             {
248                 printf("%d -> ", cur -> data);
249                 cur = cur-> link;
250             }
251         }
252     }
253
254     void search_node(NODE head)
255     {
256         int status=0, key;
257         NODE cur;
258         printf("Enter the number to search\n");
259         scanf("%d", &key);
260         if(head==NULL)
261         {
262             printf("Singly linked list empty\n");
263         }
264         else
265         {
266             cur = head;
267             while (cur != NULL)
268             {
269                 if(cur -> data == key)
270                 {
271                     status=1;
272                     break;
273                 }
274                  cur = cur->link;
275             }
276             if(status==1)
277             {
278                 printf("Search successfull, %d found\n", key);
279             }
280             else
281             {
282                 printf("Search unsuccessfull, %d not found\n", key);
283             }
284         }
285     }
286
287     NODE search_and_delete(NODE head)
288     {
289         int status=0, key;
290         NODE cur, prev;
291         printf("Enter a number\n");
292         scanf("%d", &key);
293         if(head==NULL)
294         {
295             printf("Singly linked list empty\n");
296         }
297         else if(head->data == key)
298         {
299             head = delete_front(head);
300         }
301         else
302         {
303             prev=NULL;
304             cur = head;
305             while (cur != NULL)
306             {
307                 if(cur -> data == key)
```

```
            {
                status=1;
                break;
            }
            prev=cur;
            cur = cur->link;
        }
        if(status==1)
        {
            prev->link = cur->link;
            printf("Deleted %d\n", cur->data);
            free(cur);
        }
        else
        {
            printf("%d not found\n", key);
        }
    }
    return head;
}


NODE delete_position(NODE head)
{
    int pos;
    NODE cur, prev;
    int count = count_nodes(head);
    printf("Enter position to delete\n");
    scanf("%d", &pos);
    if(head==NULL)
    {
        printf("Singly linked list empty\n");
    }
    else if(pos==1 && head->link == NULL)
    {
        printf("Deleted %d\n", head->data);
        free(head);
        head=NULL;
    }
    else if(pos==1)
    {
        cur = head;
        head = head->link;
        cur->link = NULL;
        printf("Deleted %d\n", cur->data);
        free(cur);
    }
    else if(pos==count)
    {
        head = delete_end(head);
    }
    else if(pos>1 && pos<count)
    {
        prev=NULL;
        cur=head;
        for(int i =1 ; i<pos; i++)
        {
            prev = cur;
            cur = cur->link;
        }
        prev->link = cur->link;
        printf("Deleted %d \n", cur->data);
        free(cur);
    }
    else
    {
        printf("Invalid position\n");
    }
    return head;
}

```