```c
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int data;
        struct node* next;
};
struct node* head = NULL;

int size(void)
{
        struct node* temp = head;
        if(temp == NULL)
                return 0;
        else
        {
                int count = 1;
                while(temp->next != head)
                {
                        count++;
                        temp = temp->next;
                }
                return count;
        }
}

void insert(void)
{
        struct node* new_node = (struct node*)malloc(sizeof(struct node));
        printf("Enter data\n");
        scanf("%d", &new_node->data);
        new_node->next = NULL;
        int n;
        if(head == NULL)
        {
                head = new_node;
                new_node->next = head;
                return;
        }
        struct node* temp = head;
        printf("position?\n");
        scanf("%d", &n);
        if(n == 1)
        {
                struct node* temp = head;
                while(temp->next != head)
                        temp = temp->next;
                temp->next = new_node;
                new_node->next = head;
                head = new_node;
        }
        else if(n<size()+1)
        {
                struct node* p = head;
                struct node* q = head;
                int i = 0;
```

```
56                  int i = 0;
57                  for(i=0; i<n-2; i++)
58                          p = p->next;
59                  q = p->next;
60                  new_node->next = q;
61                  p->next = new_node;
62          }
63          else
64          {
65                  struct node* temp = head;
66                  while(temp->next != head)
67                          temp = temp->next;
68                  temp->next = new_node;
69                  new_node->next = head;
70          }
71  }
72
73  void display(void)
74  {
75          struct node* temp = head;
76          while(temp->next!=head)
77          {
78                  printf("%d\t", temp->data);
79                  temp = temp->next;
80          }
81          printf("%d\n", temp->data);
82          temp = head;
83  }
84
85  void delete1()
86  {
87          struct node *p = head, *q = head, *temp = head;
88          int n, i;
89          printf("position?\n");
90          scanf("%d", &n);
91          if(n==1 && size()!=1)
92          {
93                  while(temp->next != head)
94                          temp = temp->next;
95                  temp->next = head->next;
96                  q = p->next;
97                  head = q;
98                  p->next = NULL;
99                  free(p);
100         }
101         else if(n==1 && size()==1)
102         {
103                 free(p);
104                 head = NULL;
105         }
106         else if(n<size())
107         {
108                 for(i=0; i<n-2; i++)
109                         p = p->next;
110                 q = p->next;
111                 p->next = q->next;
```

```
111                 p->next = q->next;
112                 q->next = NULL;
113                 free(q);
114         }
115         else
116         {
117                 for(i=0; i<size()-2; i++)
118                         p = p->next;
119                 q = p->next;
120                 q->next = NULL;
121                 p->next = head;
122                 free(q);
123         }
124 }
125
126 int main()
127 {
128         int choice;
129         while(1)
130         {
131                 printf("1. insert\t2. display\t3. Delete\n");
132                 scanf("%d", &choice);
133                 switch(choice)
134                 {
135                         case 1:
136                                 insert();
137                                 break;
138                         case 2:
139                                 display();
140                                 break;
141                         case 3:
142                                 delete1();
143                                 break;
144                         default:
145                                 exit(1);
146                                 break;
147                 }
148         }
149         return 0;
150 }
```