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

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

struct Node *head = NULL;

void insert();
void begin();
void end();
void atanypos();
void display();

int main()
{
    int choice;
    while (1)
    {
        printf("\n1.Insert elements\n");
        printf("2.Delete at the beginning\n");
        printf("3.Delete at the end\n");
        printf("4.Delete at any position\n");
        printf("5.Display\n");
        printf("6.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            insert();
            break;
        case 2:
            begin();
            break;
        case 3:
            end();
            break;
        case 4:
            atanypos();
            break;
```

```
        case 5:
            display();
            break;
        case 6:
            exit(0);
        default:
            printf("Invalid choice\n");
        }
    }
    return 0;
}

void insert()
{
    struct Node *ptr, *temp;
    int data;
    ptr = (struct Node *)malloc(sizeof(struct Node));
    if (ptr == NULL)
    {
        printf("Memory allocation failed\n");
        exit(1);
    }
    printf("Enter data to insert: ");
    scanf("%d", &data);
    ptr->data = data;
    ptr->next = NULL;

    if (head == NULL)
    {
        head = ptr;
        printf("Node inserted successfully\n");
    }
    else
    {
        temp = head;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = ptr;
        printf("Node inserted successfully\n");
    }
}
```

```
void begin()
{
   struct Node *ptr;
   if (head == NULL)
   {
      printf("List is empty\n");
   }
   else
   {
      ptr = head;
      head = ptr->next;
      free(ptr);
      printf("Node deleted from the beginning\n");
   }
}

void end()
{
   struct Node *ptr, *prev;
   if (head == NULL)
   {
      printf("List is empty\n");
   }
   else
   {
      ptr = head;
      while (ptr->next != NULL)
      {
         prev = ptr;
         ptr = ptr->next;
      }
      if (ptr == head)
      {
         head = NULL;
      }
      else
      {
         prev->next = NULL;
      }
      free(ptr);
      printf("Node deleted from the end\n");
   }
}
```

```
void atanypos()
{
    struct Node *ptr, *prev;
    int loc, i = 1;
    printf("Enter the position: ");
    scanf("%d", &loc);
    if (head == NULL)
    {
        printf("List is empty\n");
        return;
    }
    ptr = head;
    if (loc == 1)
    {
        head = ptr->next;
        free(ptr);
        printf("Node deleted from position %d\n", loc);
        return;
    }
    while (ptr != NULL && i < loc)
    {
        prev = ptr;
        ptr = ptr->next;
        i++;
    }
    if (ptr == NULL)
    {
        printf("Position %d is out of bounds\n", loc);
    }
    else
    {
        prev->next = ptr->next;
        free(ptr);
        printf("Node deleted from position %d\n", loc);
    }
}

void display()
{
    struct Node *ptr;
    if (head == NULL)
    {
        printf("List is empty\n");
    }
```

```
    else
    {
        ptr = head;
        while (ptr != NULL)
        {
            printf("%d ", ptr->data);
            ptr = ptr->next;
        }
        printf("\n");
    }
}
```

```
1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 12
Node inserted successfully

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 13
Node inserted successfully

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 14
Node inserted successfully

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 15
Node inserted successfully

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 5
12 13 14 15
```

```
1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 2
Node deleted from the beginning

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 5
13 14 15

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 16
Node inserted successfully

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 17
Node inserted successfully

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 18
Node inserted successfully
```

```
1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 1
Enter data to insert: 18
Node inserted successfully

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 5
13 14 15 16 17 18

1.Insert elements
2.Delete at the beginning
3.Delete at the end
4.Delete at any position
5.Display
6.Exit
Enter your choice: 3
Node deleted from the end
```