Write a Circular Link List code with functions perform on it.

Code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;

struct node
{
 int info;
 struct node *next;
}*last;

class circular_llist
{
 public:
 void create_node(int value);
 void add_begin(int value);
 void add_after(int value, int position);
 void delete_element(int value);
 void search_element(int value);
 void display_list();
 void update();
 void sort();
 circular_llist()
 {
 last = NULL;
 }
};
void circular_llist::create_node(int value)
{
```

```
    struct node *temp;
    temp = new(struct node);
    temp->info = value;
    if (last == NULL)
    {
        last = temp;
        temp->next = last;
    }
    else
    {
        temp->next = last->next;
        last->next = temp;
        last = temp;
    }
}
void circular_llist::add_begin(int value)
{
 if (last == NULL)
 {
 cout<<"First Create the list."<<endl;
 return;
 }
 struct node *temp;
 temp = new(struct node);
 temp->info = value;
 temp->next = last->next;
 last->next = temp;
}
void circular_llist::add_after(int value, int pos)
{
    if (last == NULL)
    {
```

```
        cout<<"First Create the list."<<endl;
        return;
    }
    struct node *temp, *s;
    s = last->next;
    for (int i = 0;i < pos-1;i++)
    {
        s = s->next;
        if (s == last->next)
        {
            cout<<"There are less than ";
            cout<<pos<<" in the list"<<endl;
            return;
        }
    }
    temp = new(struct node);
    temp->next = s->next;
    temp->info = value;
    s->next = temp;
    if (s == last)
    {
        last=temp;
    }
}

void circular_llist::delete_element(int value)
{
    struct node *temp, *s;
    s = last->next;
    if (last->next == last && last->info == value)
    {
        temp = last;
```

```
    last = NULL;
    free(temp);
    return;
}
if (s->info == value)
{
    temp = s;
    last->next = s->next;
    free(temp);
    return;
}
while (s->next != last)
{  /*Deletion of Element in between*/
    if (s->next->info == value)
    {
        temp = s->next;
        s->next = temp->next;
        free(temp);
        cout<<"Element "<<value;
        cout<<" deleted from the list"<<endl;
        return;
    }
    s = s->next;
}

if (s->next->info == value)
{
    temp = s->next;
    s->next = last->next;
    free(temp);
    last = s;
    return;
```

```
    }
    cout<<"Element "<<value<<" not found in the list"<<endl;
}



void circular_llist::search_element(int value)
{
    struct node *s;
    int counter = 0;
    s = last->next;
    while (s != last)
    {
        counter++;
        if (s->info == value)
        {
            cout<<"Element "<<value;
            cout<<" found at position "<<counter<<endl;
            return;
        }
        s = s->next;
    }
    if (s->info == value)
    {
        counter++;
        cout<<"Element "<<value;
        cout<<" found at position "<<counter<<endl;
        return;
    }
    cout<<"Element "<<value<<" not found in the list"<<endl;
}

void circular_llist::display_list()
```

```
{
   struct node *s;
   if (last == NULL)
   {
    cout<<"List is empty, nothing to display"<<endl;
    return;
   }
   s = last->next;
   cout<<"Circular Link List: "<<endl;
   while (s != last)
   {
      cout<<s->info<<"->";
      s = s->next;
   }
   cout<<s->info<<endl;
}



void circular_llist::sort()
{
   struct node *s, *ptr;
   int temp;
   if (last == NULL)
   {
      cout<<"List is empty, nothing to sort"<<endl;
      return;
   }
    s = last->next;
   while (s != last)
   {
      ptr = s->next;
```

```
        while (ptr != last->next)
        {
            if (ptr != last->next)
            {
                if (s->info > ptr->info)
                {
                    temp = s->info;
                    s->info = ptr->info;
                    ptr->info = temp;
                }
            }
                else
            {
                    break;
            }
            ptr = ptr->next;
        }
        s = s->next;
    }
}

int main()
{
    int choice, element, position;
    circular_llist cl;
    while (1)
    {
        cout<<endl<<"----------------------------"<<endl;
        cout<<endl<<"Circular singly linked list"<<endl;
        cout<<endl<<"----------------------------"<<endl;
        cout<<"1.Create Node"<<endl;
        cout<<"2.Add at beginning"<<endl;
```

```
cout<<"3.Add after"<<endl;
cout<<"4.Delete"<<endl;
cout<<"5.Search"<<endl;
cout<<"6.Display"<<endl;
cout<<"7.Sort"<<endl;
cout<<"8.Quit"<<endl;
cout<<"Enter your choice : ";
cin>>choice;
switch(choice)
{
   case 1:
      cout<<"Enter the element: ";
      cin>>element;
      cl.create_node(element);
      cout<<endl;
      break;
   case 2:
      cout<<"Enter the element: ";
      cin>>element;
      cl.add_begin(element);
      cout<<endl;
      break;
   case 3:
      cout<<"Enter the element: ";
      cin>>element;
      cout<<"Insert element after position: ";
      cin>>position;
      cl.add_after(element, position);
      cout<<endl;
      break;
   case 4:
      if (last == NULL)
```

```
        {
            cout<<"List is empty, nothing to delete"<<endl;
            break;
        }
        cout<<"Enter the element for deletion: ";
        cin>>element;
        cl.delete_element(element);
        cout<<endl;
        break;
    case 5:
        if (last == NULL)
        {
            cout<<"List Empty!! Can't search"<<endl;
            break;
        }
        cout<<"Enter the element to be searched: ";
        cin>>element;
        cl.search_element(element);
        cout<<endl;
        break;
    case 6:
        cl.display_list();
        break;
    case 7:
        cl.sort();
        break;
    case 8:
        exit(1);
        break;
    default:
        cout<<"Wrong choice"<<endl;
}
```

```
    }
    return 0;
}
```

```
---------------------------
Circular singly linked list
---------------------------
1.Create Node
2.Add at beginning
3.Add after
4.Delete
5.Search
6.Display
7.Sort
8.Quit
Enter your choice : 1
Enter the element: 12
```

```
---------------------------
Circular singly linked list
---------------------------
1.Create Node
2.Add at beginning
3.Add after
4.Delete
5.Search
6.Display
7.Sort
8.Quit
Enter your choice : 2
Enter the element: 14
```

```
---------------------------
Circular singly linked list
---------------------------
1.Create Node
2.Add at beginning
3.Add after
4.Delete
5.Search
6.Display
7.Sort
8.Quit
Enter your choice : 2
Enter the element: 8
```

```
---------------------------
Circular singly linked list
---------------------------
1.Create Node
2.Add at beginning
3.Add after
4.Delete
5.Search
6.Display
7.Sort
8.Quit
Enter your choice : 4
Enter the element for deletion: 12
```

```
---------------------------
Circular singly linked list
---------------------------
1.Create Node
2.Add at beginning
3.Add after
4.Delete
5.Search
6.Display
7.Sort
8.Quit
Enter your choice : 6
Circular Link List:
17->8->14
```