```
 1: /*
 2:  * C++ Program to Implement Circular Linked List
 3:  */
 4: #include<iostream>
 5: #include<cstdio>
 6: #include<cstdlib>
 7: using namespace std;
 8: /*
 9:  * Node Declaration
10:  */
11: struct node
12: {
13:     int info;
14:     struct node *next;
15: }*last;
16:
17: /*
18:  * Class Declaration
19:  */
20: class circular_llist
21: {
22:     public:
23:         void create_node(int value);
24:         void add_begin(int value);
25:         void add_after(int value, int position);
26:         void delete_element(int value);
27:         void search_element(int value);
28:         void display_list();
29:         void update();
30:         void sort();
31:         circular_llist()
32:         {
33:             last = NULL;
34:         }
35: };
```

```
36:
37: /*
38:  * Main :contains menu
39:  */
40: int main()
41: {
42:     int choice, element, position;
43:     circular_llist cl;
44:     while (1)
45:     {
46:         cout<<endl<<"----------------------------"<<endl;
47:         cout<<endl<<"Circular singly linked list"<<endl;
48:         cout<<endl<<"----------------------------"<<endl;
49:         cout<<"1. Create Node"<<endl;
50:         cout<<"2. Add at beginning"<<endl;
51:         cout<<"3. Add after"<<endl;
52:         cout<<"4. Delete"<<endl;
53:         cout<<"5. Search"<<endl;
54:         cout<<"6. Display"<<endl;
55:         cout<<"7. Update"<<endl;
56:         cout<<"8. Sort"<<endl;
57:         cout<<"9. Quit"<<endl;
58:         cout<<"Enter your choice : ";
59:         cin>>choice;
60:         switch(choice)
61:         {
62:         case 1:
63:             cout<<"Enter the element: ";
64:             cin>>element;
65:             cl.create_node(element);
66:             cout<<endl;
67:             break;
68:         case 2:
69:             cout<<"Enter the element: ";
70:             cin>>element;
```

```
71:                 cl.add_begin(element);
72:                 cout<<endl;
73:                 break;
74:             case 3:
75:                 cout<<"Enter the element: ";
76:                 cin>>element;
77:                 cout<<"Insert element after position: ";
78:                 cin>>position;
79:                 cl.add_after(element, position);
80:                 cout<<endl;
81:                 break;
82:             case 4:
83:                 if (last == NULL)
84:                 {
85:                     cout<<"List is empty, nothing to delete"<<endl
86:                     break;
87:                 }
88:                 cout<<"Enter the element for deletion: ";
89:                 cin>>element;
90:                 cl.delete_element(element);
91:                 cout<<endl;
92:                 break;
93:             case 5:
94:                 if (last == NULL)
95:                 {
96:                     cout<<"List Empty!! Can't search"<<endl;
97:                     break;
98:                 }
99:                 cout<<"Enter the element to be searched: ";
100:                cin>>element;
101:                cl.search_element(element);
102:                cout<<endl;
103:                break;
104:            case 6:
105:                cl.display_list();
```

```
106:                 break;
107:             case 7:
108:                 cl.update();
109:                 break;
110:             case 8:
111:                 cl.sort();
112:                 break;
113:             case 9:
114:                 exit(1);
115:                 break;
116:             default:
117:                 cout<<"Wrong choice"<<endl;
118:             }
119:     }
120:     return 0;
121:}
122:
123:/*
124: * Create Circular Link List
125: */
126:void circular_llist::create_node(int value)
127:{
128:     struct node *temp;
129:     temp = new(struct node);
130:     temp->info = value;
131:     if (last == NULL)
132:     {
133:         last = temp;
134:         temp->next = last;
135:     }
136:     else
137:     {
138:         temp->next = last->next;
139:         last->next = temp;
140:         last = temp;
```

```
141:     }
142:}
143:
144:/*
145: * Insertion of element at beginning
146: */
147:void circular_llist::add_begin(int value)
148:{
149:     if (last == NULL)
150:     {
151:          cout<<"First Create the list."<<endl;
152:          return;
153:     }
154:     struct node *temp;
155:     temp = new(struct node);
156:     temp->info = value;
157:     temp->next = last->next;
158:     last->next = temp;
159:}
160:
161:/*
162: * Insertion of element at a particular place
163: */
164:void circular_llist::add_after(int value, int pos)
165:{
166:     if (last == NULL)
167:     {
168:          cout<<"First Create the list."<<endl;
169:          return;
170:     }
171:     struct node *temp, *s;
172:     s = last->next;
173:     for (int i = 0;i < pos-1;i++)
174:     {
175:          s = s->next;
```

```
176:         if (s == last->next)
177:         {
178:             cout<<"There are less than ";
179:             cout<<pos<<" in the list"<<endl;
180:             return;
181:         }
182:     }
183:     temp = new(struct node);
184:     temp->next = s->next;
185:     temp->info = value;
186:     s->next = temp;
187:     /*Element inserted at the end*/
188:     if (s == last)
189:     {
190:         last=temp;
191:     }
192:}
193:
194:/*
195: * Deletion of element from the list
196: */
197:void circular_llist::delete_element(int value)
198:{
199:     struct node *temp, *s;
200:     s = last->next;
201:       /* If List has only one element*/
202:     if (last->next == last && last->info == value)
203:     {
204:         temp = last;
205:         last = NULL;
206:         free(temp);
207:         return;
208:     }
209:     if (s->info == value)  /*First Element Deletion*/
210:     {
```

```
211:          temp = s;
212:          last->next  = s->next;
213:          free(temp);
214:          return;
215:      }
216:  *last;
217:      {
218:          /*Deletion of Element in between*/
219:          if (s->next->info == value)
220:          {
221:              temp = s->next;
222:              s->next  = temp->next;
223:              free(temp);
224:              cout<<"Element "<<value;
225:              cout<<" deleted from the list"<<endl;
226:              return;
227:          }
228:          s = s->next;
229:      }
230:      /*Deletion of last element*/
231:      if (s->next->info == value)
232:      {
233:          temp = s->next;
234:          s->next  = last->next;
235:          free(temp);
236:          last = s;
237:          return;
238:      }
239:      cout<<"Element "<<value<<" not found in the list"<<endl;
240:}
241:
242:/*
243:  * Search element in the list
244:  */
245:void circular_llist::search_element(int value)
```

```
246:{
247:     struct node *s;
248:     int counter = 0;
249:     s = last->next;
250:     while (s != last)
251:     {
252:          counter++;
253:          if (s->info == value)
254:          {
255:               cout<<"Element "<<value;
256:               cout<<" found at position "<<counter<<endl;
257:               return;
258:          }
259:          s = s->next;
260:     }
261:     if (s->info == value)
262:     {
263:          counter++;
264:          cout<<"Element "<<value;
265:          cout<<" found at position "<<counter<<endl;
266:          return;
267:     }
268:     cout<<"Element "<<value<<" not found in the list"<<endl;
269:}
270:
271:/*
272: * Display Circular Link List
273: */
274:void circular_llist::display_list()
275:{
276:     struct node *s;
277:     if (last == NULL)
278:     {
279:          cout<<"List is empty, nothing to display"<<endl;
280:          return;
```

```
281:     }
282:     s = last->next;
283:     cout<<"Circular Link List: "<<endl;
284:     while (s != last)
285:     {
286:         cout<<s->info<<"->";
287:         s = s->next;
288:     }
289:     cout<<s->info<<endl;
290:}
291:
292:/*
293: * Update Circular Link List
294: */
295:void circular_llist::update()
296:{
297:     int value, pos, i;
298:     if (last == NULL)
299:     {
300:         cout<<"List is empty, nothing to update"<<endl;
301:         return;
302:     }
303:     cout<<"Enter the node position to be updated: ";
304:     cin>>pos;
305:     cout<<"Enter the new value: ";
306:     cin>>value;
307:     struct node *s;
308:     s = last->next;
309:     for (i = 0;i < pos - 1;i++)
310:     {
311:         if (s == last)
312:         {
313:             cout<<"There are less than "<<pos<<" elements.";
314:             cout<<endl;
315:             return;
```

```
316:         }
317:         s = s->next;
318:     }
319:     s->info = value;
320:     cout<<"Node Updated"<<endl;
321:}
322:
323:/*
324: * Sort Circular Link List
325: */
326:void circular_llist::sort()
327:{
328:     struct node *s, *ptr;
329:     int temp;
330:     if (last == NULL)
331:     {
332:         cout<<"List is empty, nothing to sort"<<endl;
333:         return;
334:     }
335:     s = last->next;
336:     while (s != last)
337:     {
338:         ptr = s->next;
339:         while (ptr != last->next)
340:         {
341:             if (ptr != last->next)
342:             {
343:                 if (s->info > ptr->info)
344:                 {
345:                     temp = s->info;
346:                     s->info = ptr->info;
347:                     ptr->info = temp;
348:                 }
349:             }
350:             else
```

```
351:                {
352:                    break;
353:                }
354:                ptr = ptr->next;
355:            }
356:            s = s->next;
357:        }
358:}
```