

PRN: 2020BTEIT00041

Name: Om Vivek Gharge

Q.) Circular Linked List

CODE:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  class Node{
5  public:
6      int data;
7      Node *next;
8
9      Node(int d){
10         data = d;
11         next = NULL;
12     }
13 };
14
15 class Head{
16 public:
17     int count;
18     Node *first;
19     Node *last;
20
21     Head(int c, Node *f, Node* l){
22         this->count = c;
23         this->first = f;
24         this->last = l;
25     }
26 };
27
28 // Function to insert at head of circular linked list
29 void insertAtHead(Head *head, int data){
30     Node *newNode = new Node(data);
31
32     if(head->count == 0){
33         head->first = newNode;
34         head->last = newNode;
35
36         head->first->next = head->last;
37         head->last->next = head->first;
38     }
39     else{
```

```

40     newNode->next = head->first;
41     head->first = newNode;
42     head->last->next = head->first;
43 }
44 head->count++;
45 }
46
47 // Function to insert at tail of circular linked list
48 void insertAtTail(Head *head, int data){
49     Node *newNode = new Node(data);
50
51     if(head->count == 0){
52         head->first = newNode;
53         head->last = newNode;
54
55         head->first->next = head->last;
56         head->last->next = head->first;
57     }
58     else{
59         head->last->next = newNode;
60         head->last = newNode;
61         head->last->next = head->first;
62     }
63     head->count++;
64 }
65
66 // Function to insert at any position of circular linked list
67 void insertAfter(Head *head, int data, int pos){
68     Node *newNode = new Node(data);
69     Node *temp = head->first;
70     Node *prev = head->first;
71
72     if(pos == 1){
73         insertAtHead(head, data);
74     }
75     else if(pos == head->count + 1){
76         insertAtTail(head, data);

```

```

77     }
78     else{
79         for(int i = 1; i < pos; i++){
80             prev = temp;
81             temp = temp->next;
82         }
83         prev->next = newNode;
84         newNode->next = temp;
85         head->count++;
86     }
87 }
88
89 // Function to delete a node by value from circular linked list
90 void deletebyValue(Head *head, int data){
91     Node *temp = head->first;
92     Node *prev = head->first;
93
94     if(head->count == 0){
95         cout << "List is empty" << endl;
96     }
97     else if(head->count == 1){
98         if(head->first->data == data){
99             head->first = NULL;
100             head->last = NULL;
101             head->count--;
102         }
103         else{
104             cout << "Element not found" << endl;
105         }
106     }
107     else{
108         if(head->first->data == data){
109             head->first = head->first->next;
110             head->last->next = head->first;
111             head->count--;
112         }
113         else{
114             while(temp->data != data){

```

```

115         prev = temp;
116         temp = temp->next;
117     }
118     if(temp->data == data){
119         prev->next = temp->next;
120         head->count--;
121     }
122     else{
123         cout << "Element not found" << endl;
124     }
125 }
126 }
127 }
128
129 // Function to delete a node by position from circular linked list
130 void deleteByPos(Head *head, int pos){
131     Node *temp = head->first;
132     Node *prev = head->first;
133
134     if(head->count == 0){
135         cout << "List is empty" << endl;
136     }
137     else if(head->count == 1){
138         if(pos == 1){
139             head->first = NULL;
140             head->last = NULL;
141             head->count--;
142         }
143         else{
144             cout << "Element not found" << endl;
145         }
146     }
147     else{
148         if(pos == 1){
149             head->first = head->first->next;
150             head->last->next = head->first;
151             head->count--;

```

```

152     }
153     else if(pos == head->count){
154         while(temp->next != head->first){
155             prev = temp;
156             temp = temp->next;
157         }
158         prev->next = head->first;
159         head->last = prev;
160         head->count--;
161     }
162     else{
163         for(int i = 1; i < pos; i++){
164             prev = temp;
165             temp = temp->next;
166         }
167         prev->next = temp->next;
168         head->count--;
169     }
170 }
171 }
172
173 // Function to search a node by value from circular linked list
174 void searchByValue(Head *head, int data){
175     Node *temp = head->first;
176
177     if(head->count == 0){
178         cout << "List is empty" << endl;
179     }
180     else{
181         for(int i=0; i<head->count; i++){
182             if(temp->data == data){
183                 cout << "Element found at position " << i+1 << endl;
184                 return;
185             }
186             else{
187                 temp = temp->next;
188             }
189         }
190     }

```

```

191 }
192
193 // Function to print circular linked list
194 void printList(Head *head){
195     Node *temp = head->first;
196
197     if(head->count == 0){
198         cout << "List is empty" << endl;
199     }
200     else{
201         for(int i=0; i<head->count; i++){
202             cout << temp->data << " ";
203             temp = temp->next;
204         }
205         cout << endl;
206     }
207 }
208
209 int main(){
210     Head* h = new Head(0, NULL, NULL);
211
212     int opt, data, index;
213     char choice;
214     while(1){
215         cout<<"\nMENU\n a. Add directly to LinkedList.\n b. Use functions.\n";
216         cout<<"Choose: ";
217         cin>>choice;
218
219         switch (choice){
220
221             case 'a':
222                 int num;
223                 cout<<"Enter the data to add in LinkedList (enter '-1' if you want stop): ";
224                 while(1){
225                     cin>>num;
226                     if(num == -1) break;

```

```

227
228     Node* addNode = new Node(num);
229     if(h->count == 0){
230         h->first = addNode;
231         h->last = addNode;
232         addNode->next = h->first;
233
234         h->count++;
235     }
236     else{
237         h->last->next = addNode;
238         h->last = addNode;
239         addNode->next = h->first;
240
241         h->count++;
242     }
243 }
244 break;
245
246 case 'b':
247     cout<<"\nMENU 1.0\n 1.Add at head\n 2.Add at tail\n 3.Add after\n 4.Delete (by Value)\n 5.Delete (by Index))\n 6.Search\n 7.Display the List\n 8.Exit\n";
248     cout<<"Enter your option: ";
249
250     cin>>opt;
251     cout<<"\n";
252     if(opt>7) break;
253
254     switch(opt){
255
256         case 1:
257             cout<<"Enter data to add: ";
258             cin>>data;
259             cout<<"Adding data...\n";
260             insertAtHead(h, data);
261             cout<<"\n";
262             break;
263
264         case 2:
265             cout<<"Enter data to add: ";
266             cin>>data;
267             cout<<"Adding data...\n";
268             insertAtTail(h, data);
269             cout<<"\n";
270             break;
271
272         case 3:
273             cout<<"Enter the index: ";
274             cin>>index;
275             cout<<"Enter data to add: ";
276             cin>>data;
277             cout<<"Adding data...\n";
278             insertAfter(h, data, index);
279             cout<<"\n";
280             break;
281
282         case 4:
283             cout<<"Enter data to delete: ";
284             cin>>data;
285             cout<<"Deleting data...\n";
286             deleteByValue(h, data);
287             cout<<"\n";
288             break;
289
290         case 5:
291             cout<<"Enter the index: ";
292             cin>>index;
293             cout<<"Deleting data...\n";
294             deleteByPos(h, index);
295             cout<<"\n";
296             break;
297
298         case 6:
299             cout<<"Enter data to search: ";
300             cin>>data;
301             cout<<"Searching data...\n";
302             searchByValue(h, data);
303             cout<<endl;

```

```
304         break;
305
306     case 7:
307         cout<<"Displaying the LinkedList: ";
308         printList(h);
309         cout<<"\n";
310         break;
311
312     default:
313         break;
314
315     }
316
317     default:
318         break;
319     }
320 }
321
322 return 0;
323 }
```


OUTPUT:

```
MENU
a. Add directly to LinkedList.
b. Use functions.
Choose: a
Enter the data to add in LinkedList (enter '-1' if you want stop): 1 2 3 4 5
-1

MENU
a. Add directly to LinkedList.
b. Use functions.
Choose: b

MENU 1.0
1.Add at head
2.Add at tail
3.Add after
4.Delete (by Value)
5.Delete (by Index))
6.Search
7.Display the List
8.Exit
Enter your option: 1

Enter data to add: 0
Adding data...

MENU
a. Add directly to LinkedList.
b. Use functions.
Choose: b

MENU 1.0
1.Add at head
2.Add at tail
3.Add after
4.Delete (by Value)
5.Delete (by Index))
6.Search
7.Display the List
8.Exit
Enter your option: 7
```

```
Displaying the LinkedList: 0 1 2 3 4 5
```

```
MENU
```

- a. Add directly to LinkedList.
- b. Use functions.

```
Choose: b
```

```
MENU 1.0
```

- 1.Add at head
- 2.Add at tail
- 3.Add after
- 4.Delete (by Value)
- 5.Delete (by Index))
- 6.Search
- 7.Display the List
- 8.Exit

```
Enter your option: 4
```

```
Enter data to delete: 5
```

```
Deleting data...
```

```
MENU
```

- a. Add directly to LinkedList.
- b. Use functions.

```
Choose: b
```

```
MENU 1.0
```

- 1.Add at head
- 2.Add at tail
- 3.Add after
- 4.Delete (by Value)
- 5.Delete (by Index))
- 6.Search
- 7.Display the List
- 8.Exit

```
Enter your option: 7
```

```
Displaying the LinkedList: 0 1 2 3 4
```

ALGORITHM:

Circular Linked List

ALGORITHMS:

1.Insertion of node in the circular linked list at the beginning

Step 1:

IF PTR = NULL

Write OVERFLOW

Go to Step 11

[END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW_NODE -> DATA = VAL

Step 5: SET TEMP = HEAD

Step 6: Repeat Step 8 while TEMP -> NEXT != HEAD

Step 7: SET TEMP = TEMP -> NEXT

[END OF LOOP]

Step 8: SET NEW_NODE -> NEXT = HEAD

Step 9: SET TEMP -> NEXT = NEW_NODE

Step 10: SET HEAD = NEW_NODE

Step 11: EXIT

2. Insertion of node in the circular linked list at the end

Step 1: IF PTR = NULL

Write OVERFLOW

Go to Step 1

[END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW_NODE -> DATA = VAL

Step 5: SET NEW_NODE -> NEXT = HEAD

Step 6: SET TEMP = HEAD

Step 7: Repeat Step 8 while TEMP -> NEXT != HEAD

Step 8: SET TEMP = TEMP -> NEXT

[END OF LOOP]

Step 9: SET TEMP -> NEXT = NEW_NODE

Step 10: EXIT

3. Deletion of node in the circular linked list at the beginning

Step 1: IF HEAD = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = HEAD

Step 3: Repeat Step 4 while PTR → NEXT != HEAD

Step 4: SET PTR = PTR → next

[END OF LOOP]

Step 5: SET PTR → NEXT = HEAD → NEXT

Step 6: FREE HEAD

Step 7: SET HEAD = PTR → NEXT

Step 8: EXIT

4. Deletion of node in the circular linked list at the end

Step 1: IF HEAD = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = HEAD

Step 3: Repeat Steps 4 and 5 while PTR -> NEXT != HEAD

Step 4: SET PREPTR = PTR

Step 5: SET PTR = PTR -> NEXT

[END OF LOOP]

Step 6: SET PREPTR -> NEXT = HEAD

Step 7: FREE PTR

Step 8: EXIT

5. Searching in circular linked list

Step 1: SET PTR = HEAD

Step 2: Set I = 0

STEP 3: IF PTR = NULL

WRITE "EMPTY LIST"

GOTO STEP 8

END OF IF

STEP 4: IF HEAD \rightarrow DATA = ITEM

WRITE i+1 RETURN [END OF IF]

STEP 5: REPEAT STEP 5 TO 7 UNTIL PTR \rightarrow next \neq head

STEP 6: if ptr \rightarrow data = item

write i+1

RETURN

End of IF

STEP 7: I = I + 1

STEP 8: PTR = PTR \rightarrow NEXT

[END OF LOOP]

STEP 9: EXIT

6.Traversing in circular linked list

STEP 1: SET PTR = HEAD

STEP 2: IF PTR = NULL

WRITE "EMPTY LIST"

GOTO STEP 8

END OF IF

STEP 4: REPEAT STEP 5 AND 6 UNTIL PTR \rightarrow NEXT \neq HEAD

STEP 5: PRINT PTR \rightarrow DATA

STEP 6: PTR = PTR \rightarrow NEXT

[END OF LOOP]

STEP 7: PRINT PTR \rightarrow DATA

STEP 8: EXIT