**CIRCULAR SINGLY LINKED LIST**

1. #include<stdio.h>
2. #include<stdlib.h>
3. struct node
4. {
5. **int** data;
6. struct node \*next;
7. };
8. struct node \*head;
9. **void** beginsert ();
10. **void** lastinsert ();
11. **void** randominsert();
12. **void** begin\_delete();
13. **void** last\_delete();
14. **void** random\_delete();
15. **void** display();
16. **void** search();
17. **void** main ()
18. {
19. **int** choice =0;
20. **while**(choice != 7)
21. {
22. printf("\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");
23. printf("\nChoose one option from the following list ...\n");
24. printf("\n===============================================\n");
25. printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from Beginning\n4.Delete from last\n5.Search for an element\n6.Show\n7.Exit\n");
26. printf("\nEnter your choice?\n");
27. scanf("\n%d",&choice);
28. **switch**(choice)
29. {
30. **case** 1:
31. beginsert();
32. **break**;
33. **case** 2:
34. lastinsert();
35. **break**;
36. **case** 3:
37. begin\_delete();
38. **break**;
39. **case** 4:
40. last\_delete();
41. **break**;
42. **case** 5:
43. search();
44. **break**;
45. **case** 6:
46. display();
47. **break**;
48. **case** 7:
49. exit(0);
50. **break**;
51. **default**:
52. printf("Please enter valid choice..");
53. }
54. }
55. }
56. **void** beginsert()
57. {
58. struct node \*ptr,\*temp;
59. **int** item;
60. ptr = (struct node \*)malloc(sizeof(struct node));
61. **if**(ptr == NULL)
62. {
63. printf("\nOVERFLOW");
64. }
65. **else**
66. {
67. printf("\nEnter the node data?");
68. scanf("%d",&item);
69. ptr -> data = item;
70. **if**(head == NULL)
71. {
72. head = ptr;
73. ptr -> next = head;
74. }
75. **else**
76. {
77. temp = head;
78. **while**(temp->next != head)
79. temp = temp->next;
80. ptr->next = head;
81. temp -> next = ptr;
82. head = ptr;
83. }
84. printf("\nnode inserted\n");
85. }
87. }
88. **void** lastinsert()
89. {
90. struct node \*ptr,\*temp;
91. **int** item;
92. ptr = (struct node \*)malloc(sizeof(struct node));
93. **if**(ptr == NULL)
94. {
95. printf("\nOVERFLOW\n");
96. }
97. **else**
98. {
99. printf("\nEnter Data?");
100. scanf("%d",&item);
101. ptr->data = item;
102. **if**(head == NULL)
103. {
104. head = ptr;
105. ptr -> next = head;
106. }
107. **else**
108. {
109. temp = head;
110. **while**(temp -> next != head)
111. {
112. temp = temp -> next;
113. }
114. temp -> next = ptr;
115. ptr -> next = head;
116. }
118. printf("\nnode inserted\n");
119. }
121. }
123. **void** begin\_delete()
124. {
125. struct node \*ptr;
126. **if**(head == NULL)
127. {
128. printf("\nUNDERFLOW");
129. }
130. **else** **if**(head->next == head)
131. {
132. head = NULL;
133. free(head);
134. printf("\nnode deleted\n");
135. }
137. **else**
138. {   ptr = head;
139. **while**(ptr -> next != head)
140. ptr = ptr -> next;
141. ptr->next = head->next;
142. free(head);
143. head = ptr->next;
144. printf("\nnode deleted\n");
146. }
147. }
148. **void** last\_delete()
149. {
150. struct node \*ptr, \*preptr;
151. **if**(head==NULL)
152. {
153. printf("\nUNDERFLOW");
154. }
155. **else** **if** (head ->next == head)
156. {
157. head = NULL;
158. free(head);
159. printf("\nnode deleted\n");
161. }
162. **else**
163. {
164. ptr = head;
165. **while**(ptr ->next != head)
166. {
167. preptr=ptr;
168. ptr = ptr->next;
169. }
170. preptr->next = ptr -> next;
171. free(ptr);
172. printf("\nnode deleted\n");
174. }
175. }
177. **void** search()
178. {
179. struct node \*ptr;
180. **int** item,i=0,flag=1;
181. ptr = head;
182. **if**(ptr == NULL)
183. {
184. printf("\nEmpty List\n");
185. }
186. **else**
187. {
188. printf("\nEnter item which you want to search?\n");
189. scanf("%d",&item);
190. **if**(head ->data == item)
191. {
192. printf("item found at location %d",i+1);
193. flag=0;
194. }
195. **else**
196. {
197. **while** (ptr->next != head)
198. {
199. **if**(ptr->data == item)
200. {
201. printf("item found at location %d ",i+1);
202. flag=0;
203. **break**;
204. }
205. **else**
206. {
207. flag=1;
208. }
209. i++;
210. ptr = ptr -> next;
211. }
212. }
213. **if**(flag != 0)
214. {
215. printf("Item not found\n");
216. }
217. }
219. }
221. **void** display()
222. {
223. struct node \*ptr;
224. ptr=head;
225. **if**(head == NULL)
226. {
227. printf("\nnothing to print");
228. }
229. **else**
230. {
231. printf("\n printing values ... \n");
233. **while**(ptr -> next != head)
234. {
236. printf("%d\n", ptr -> data);
237. ptr = ptr -> next;
238. }
239. printf("%d\n", ptr -> data);
240. }
242. }

**OUTPUT:**

