1. #include<stdio.h>
2. #include<stdlib.h>
3. struct node
4. {
5. **int** data;
6. struct node \*next;
7. };
8. struct node \*head;
10. **void** beginsert ();
11. **void** lastinsert ();
12. **void** randominsert();
13. **void** begin\_delete();
14. **void** last\_delete();
15. **void** random\_delete();
16. **void** display();
17. **void** search();
18. **void** main ()
19. {
20. **int** choice =0;
21. **while**(choice != 7)
22. {
23. printf("\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");
24. printf("\nChoose one option from the following list ...\n");
25. printf("\n===============================================\n");
26. 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");
27. printf("\nEnter your choice?\n");
28. scanf("\n%d",&choice);
29. **switch**(choice)
30. {
31. **case** 1:
32. beginsert();
33. **break**;
34. **case** 2:
35. lastinsert();
36. **break**;
37. **case** 3:
38. begin\_delete();
39. **break**;
40. **case** 4:
41. last\_delete();
42. **break**;
43. **case** 5:
44. search();
45. **break**;
46. **case** 6:
47. display();
48. **break**;
49. **case** 7:
50. exit(0);
51. **break**;
52. **default**:
53. printf("Please enter valid choice..");
54. }
55. }
56. }
57. **void** beginsert()
58. {
59. struct node \*ptr,\*temp;
60. **int** item;
61. ptr = (struct node \*)malloc(sizeof(struct node));
62. **if**(ptr == NULL)
63. {
64. printf("\nOVERFLOW");
65. }
66. **else**
67. {
68. printf("\nEnter the node data?");
69. scanf("%d",&item);
70. ptr -> data = item;
71. **if**(head == NULL)
72. {
73. head = ptr;
74. ptr -> next = head;
75. }
76. **else**
77. {
78. temp = head;
79. **while**(temp->next != head)
80. temp = temp->next;
81. ptr->next = head;
82. temp -> next = ptr;
83. head = ptr;
84. }
85. printf("\nnode inserted\n");
86. }
88. }
89. **void** lastinsert()
90. {
91. struct node \*ptr,\*temp;
92. **int** item;
93. ptr = (struct node \*)malloc(sizeof(struct node));
94. **if**(ptr == NULL)
95. {
96. printf("\nOVERFLOW\n");
97. }
98. **else**
99. {
100. printf("\nEnter Data?");
101. scanf("%d",&item);
102. ptr->data = item;
103. **if**(head == NULL)
104. {
105. head = ptr;
106. ptr -> next = head;
107. }
108. **else**
109. {
110. temp = head;
111. **while**(temp -> next != head)
112. {
113. temp = temp -> next;
114. }
115. temp -> next = ptr;
116. ptr -> next = head;
117. }
119. printf("\nnode inserted\n");
120. }
122. }
124. **void** begin\_delete()
125. {
126. struct node \*ptr;
127. **if**(head == NULL)
128. {
129. printf("\nUNDERFLOW");
130. }
131. **else** **if**(head->next == head)
132. {
133. head = NULL;
134. free(head);
135. printf("\nnode deleted\n");
136. }
138. **else**
139. {   ptr = head;
140. **while**(ptr -> next != head)
141. ptr = ptr -> next;
142. ptr->next = head->next;
143. free(head);
144. head = ptr->next;
145. printf("\nnode deleted\n");
147. }
148. }
149. **void** last\_delete()
150. {
151. struct node \*ptr, \*preptr;
152. **if**(head==NULL)
153. {
154. printf("\nUNDERFLOW");
155. }
156. **else** **if** (head ->next == head)
157. {
158. head = NULL;
159. free(head);
160. printf("\nnode deleted\n");
162. }
163. **else**
164. {
165. ptr = head;
166. **while**(ptr ->next != head)
167. {
168. preptr=ptr;
169. ptr = ptr->next;
170. }
171. preptr->next = ptr -> next;
172. free(ptr);
173. printf("\nnode deleted\n");
175. }
176. }
178. **void** search()
179. {
180. struct node \*ptr;
181. **int** item,i=0,flag=1;
182. ptr = head;
183. **if**(ptr == NULL)
184. {
185. printf("\nEmpty List\n");
186. }
187. **else**
188. {
189. printf("\nEnter item which you want to search?\n");
190. scanf("%d",&item);
191. **if**(head ->data == item)
192. {
193. printf("item found at location %d",i+1);
194. flag=0;
195. }
196. **else**
197. {
198. **while** (ptr->next != head)
199. {
200. **if**(ptr->data == item)
201. {
202. printf("item found at location %d ",i+1);
203. flag=0;
204. **break**;
205. }
206. **else**
207. {
208. flag=1;
209. }
210. i++;
211. ptr = ptr -> next;
212. }
213. }
214. **if**(flag != 0)
215. {
216. printf("Item not found\n");
217. }
218. }
220. }
222. **void** display()
223. {
224. struct node \*ptr;
225. ptr=head;
226. **if**(head == NULL)
227. {
228. printf("\nnothing to print");
229. }
230. **else**
231. {
232. printf("\n printing values ... \n");
234. **while**(ptr -> next != head)
235. {
237. printf("%d\n", ptr -> data);
238. ptr = ptr -> next;
239. }
240. printf("%d\n", ptr -> data);
241. }
243. }