

```

1  /*
2  WAP Implement Single Link List with following operations
3  a) Sort the linked list.
4  b) Reverse the linked list.
5  c) Concatenation of two linked lists
6  WAP to implement Stack & Queues using Linked Representation
7  */
8  #include <stdio.h>
9  #include <stdlib.h>
10 struct node
11 {
12     int data;
13     struct node* next;
14 };
15 struct node *rear=NULL, *front =NULL, *top=NULL;
16 struct node* getnode(int item)
17 {
18     struct node* newn = (struct node*)malloc(sizeof(struct node));
19     newn->data = item;
20     newn->next = NULL;
21     return newn;
22 }
23 void display(struct node* head)
24 {
25     if(head == NULL)
26     {
27         printf("List is empty.\n");
28         return;
29     }
30     struct node* ptr = head;
31     while(ptr)
32     {
33         printf("%d->", ptr->data);
34         ptr = ptr->next;
35     }
36     printf("\b \b\b \n");
37 }
38 struct node* insertfront(struct node* head, int item)
39 {
40     struct node* newn = getnode(item);
41     newn->next = head;

```

```

39 {
40     struct node* newn = getnode(item);
41     newn->next = head;
42     head = newn;
43     return head;
44 }
45 void swap(int *a, int *b)
46 {
47     int temp;
48     temp = *a;
49     *a = *b;
50     *b = temp;
51 }
52 struct node* sort (struct node* head)
53 {
54     int sorted;
55     if(head == NULL) return head;
56     struct node* ptr = head;
57     do
58     {
59         ptr = head;
60         sorted = 0;
61         while(ptr->next)
62         {
63             if(ptr->data > ptr->next->data)
64             {
65                 swap(&ptr->data, &ptr->next->data);
66                 sorted = 1;
67             }
68             ptr = ptr->next;
69         }
70     } while(sorted == 1);
71     return head;
72 }
73 void reverse(struct node** head)
74 {
75     struct node* prev = NULL;
76     struct node* current = *head;
77     struct node* next = NULL;
78     while (current != NULL) {
79         next = current->next;

```

```

72     }
73     void reverse(struct node** head)
74     {
75         struct node* prev = NULL;
76         struct node* current = *head;
77         struct node* next = NULL;
78         while (current != NULL) {
79             next = current->next;
80             current->next = prev;
81             prev = current;
82             current = next;
83         }
84         *head = prev;
85     }
86     struct node* concatenate(struct node* head1, struct node* head2)
87     {
88         struct node* ptr = head1;
89         while(ptr->next)
90         {
91             ptr = ptr->next;
92         }
93         ptr->next = head2;
94         return head1;
95     }
96     void qinsert()
97     {
98         struct node *newnode;
99         newnode=(struct node *) malloc(sizeof(struct node));
100         printf("Enter the element:\n");
101         scanf("%d",&newnode->data);
102         newnode->next=NULL;
103         if(rear==NULL)
104         {
105             rear=newnode;
106             front=newnode;
107         }
108         else
109         {
110             rear->next=newnode;
111             rear=newnode;
112         }

```

```

111     rear=newnode;
112 }
113 }
114 void qdel()
115 {
116     if(front==NULL)
117     {
118         printf("Queue is empty\n");return;
119     }
120     else
121     {
122         printf("Deleted ele is %d",front->data);
123         if(front==rear)
124         {
125             printf("Queue is empty\n");
126             front=NULL; rear=NULL;
127         }
128         else
129             front=front->next;
130     }
131 }
132 void qdisplay()
133 {
134     struct node *temp;
135     if(front ==NULL)
136     {
137         printf("Queue is empty");
138         return;
139     }
140     temp=front;
141     while (temp !=NULL)
142     {
143         printf("%d ",temp->data);
144         temp=temp->next;
145     }
146 }
147 void spush()
148 {
149     int item;
150     struct node *newnode;
151     printf("Enter the element\n");

```

```

151 printf("Enter the element\n");
152 scanf("%d",&item);
153 newnode=(struct node*)malloc(sizeof(struct node));
154 newnode->data=item;
155 newnode->next=NULL;
156 if(top==NULL)
157     top=newnode;
158 else
159     newnode->next=top;
160     top=newnode;
161 }
162 void spop()
163 {
164     if(top==NULL)
165         printf("stack is empty");
166     else
167     {
168         printf("element removed is %d:", top->data);
169         top=top->next;
170     }
171 }
172 void sdisplay()
173 {
174     struct node *temp;
175     temp=top;
176     if(top==NULL)
177         printf("Stack is empty");
178     while(temp!=NULL)
179     {
180         printf("%d",temp->data);
181         printf("\n");
182         temp=temp->next;
183     }
184 }
185 int main()
186 {
187     printf("Linked list program containing sort, reverse, and concatenate functions.\n");
188     int n1, n2, n, ch, flag = 0;
189     int choice;
190     struct node* head1 = NULL; struct node* head2 = NULL;
191     do

```

```

181     printf("\n");
182     temp=temp->next;
183 }
184 }
185 int main()
186 {
187     printf("Linked list program containing sort, reverse, and concatenate functions.\n");
188     int n1, n2, n, ch, flag = 0;
189     int choice;
190     struct node* head1 = NULL; struct node* head2 = NULL;
191     do
192     {
193         printf("Enter the choice\n1.Stack\n2.Queue\n3: Linked list 1\n4:Linked list 2\n5: Exit\n");
194         scanf("%d", &n1);
195         switch(n1)
196         {
197             case 1:
198             {
199                 do
200                 { printf("\n1. Push \n2. Display \n3. Pop\n");
201                     printf("\nEnter your choice : ");
202                     scanf("%d",&choice);
203                     switch(choice)
204                     {
205                         case 1: spush(); break;
206                         case 2: sdisplay();break;
207                         case 3: spop(); break;
208                     }
209                 }while(choice!=10);
210             }
211             case 2:
212             {
213                 do
214                 { printf("\nQueue implementation using linked list\n");
215                     printf("\n1. Create \n2. Display \n3. Delete \n4. Exit\n");
216                     printf("\nEnter your choice : ");
217                     scanf("%d",&choice);
218                     switch(choice)
219                     { case 1: qinsert(); break;
220                       case 2: qdisplay();break;

```



```

lab_3.c X lab_10.c X *lab8.c X lab8_ds.c X
217 printf("\nEnter your choice : ");
218 scanf("%d",&choice);
219 switch(choice)
220 { case 1: qinsert(); break;
221   case 2: qdisplay();break;
222   case 3: qdel(); break;
223 }
224 while(choice!=10);
225 }
226 case 3:
227 {
228 do
229 {
230 printf("3: Insert\n4: Sort\n5: Reverse\n6:Concatenate with list 1\n7: Display list\n8: Go back to main menu\n9:Exit\n");
231 scanf("%d", &n2);
232 switch(n2)
233 {
234 case 3: {
235 printf("Enter item to beinserted: ");
236 scanf("%d", &n);
237 head1 =
238 insertfront(head1, n);
239 break;
240 }
241 case 4: {
242 head1 = sort(head1);
243 break;
244 }
245 case 5: {
246 reverse(&head1);
247 break;
248 }
249 case 6: {
250 head1 =
251 concatenate(head1, head2);
252 break;
253 }
254 case 7: {
255 display(head1);
256 break;
257 }

```

```

Start here X lab_3.c X lab_10.c X *lab8.c X lab8_ds.c X
254 case 7: {
255     display(head1);
256     break;
257 }
258 case 8: {
259     flag = 1;
260     break;
261 }
262 case 9: {
263     exit(0);
264 }
265 default: printf("Invalid input.\n");
266 }
267 if(flag == 1)
268 {
269     break;
270 }
271 }while(1);
272 break;
273 }
274 case 4: {
275     flag = 0;
276     do
277     {
278         printf("3: Insert\n4: Sort\n5: Reverse\n6:Concatenate with list 1\n7: Display list\n8: Go back to main menu\n9:Exit\n");
279         scanf("%d", &n2);
280         switch(n2)
281         {
282             case 3: {
283                 printf("Enter item to be inserted: ");
284                 scanf("%d", &n);
285                 head2 =
286                 insertfront(head2, n);
287                 break;
288             }
289             case 4: {
290                 head2 = sort(head2);
291                 break;
292             }
293             case 5: {
294                 reverse(&head2);

```



```

288 -}
289 case 4: {
290     head2 = sort(head2);
291     break;
292 -}
293 case 5: {
294     reverse(&head2);
295     break;
296 -}
297 case 6: {
298     head2 =
299     concatenate(head2, head1);
300     break;
301 -}
302 case 7: {
303     display(head2);
304     break;
305 -}
306 case 8: {
307     flag = 1;
308     break;
309 -}
310 case 9: {
311     exit(0);
312 -}
313 default: printf("Invalid input.\n");
314 -}
315 if(flag == 1)
316 {
317     flag = 0; break;
318 -}
319 }while(1);
320 break;
321 -}
322 case 9: exit(0);
323 default: printf("Invalid input.\n");
324 -}
325 }while(1);
326 return 0;
327 }
328

```