

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  struct contact
6  {
7      char name[50];
8      char email[50];
9      char numbr[50];
10 };
11 struct linkedlist{
12     int data;
13     struct linkedlist* next;
14
15 };
16 struct Node {
17     int data;
18     struct Node* next;
19 };
20
21
22 struct Node* mergeSortedList(struct Node* list1, struct Node* list2) {
23     struct Node* mergedList = NULL;
24     struct Node* tail = NULL;
25
26     while (list1 != NULL && list2 != NULL) {
27         struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
28         if (newNode == NULL) {
29             printf("Memory allocation error\n");
30             exit(EXIT_FAILURE);
31         }
32
33         if (list1->data <= list2->data) {
34             newNode->data = list1->data;
35             list1 = list1->next;
36         } else {
37             newNode->data = list2->data;
38             list2 = list2->next;
39         }
40
41         newNode->next = NULL;
42
43         if (mergedList == NULL) {
44             mergedList = newNode;
45             tail = newNode;
46         } else {
47             tail->next = newNode;
48             tail = newNode;
49         }
50     }
51
52     // If there are remaining nodes in either list
53     if (list1 != NULL) {
54         if (mergedList == NULL) {
55             mergedList = list1;
56         } else {
57             tail->next = list1;
58         }
59     } else if (list2 != NULL) {
60         if (mergedList == NULL) {
61             mergedList = list2;
62         } else {
63             tail->next = list2;
64         }
65     }
66

```

```

67     return mergedList;
68 }
69
70 void printList(struct Node* head) {
71     while (head != NULL) {
72         printf("%d ", head->data);
73         head = head->next;
74     }
75     printf("\n");
76 }
77
78 void insertEnd(struct Node** head, int value) {
79     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
80     if (newNode == NULL) {
81         printf("Memory allocation error\n");
82         exit(EXIT_FAILURE);
83     }
84
85     newNode->data = value;
86     newNode->next = NULL;
87
88     if (*head == NULL) {
89         *head = newNode;
90     } else {
91         struct Node* temp = *head;
92         while (temp->next != NULL) {
93             temp = temp->next;
94         }
95         temp->next = newNode;
96     }
97 }
98
99 void freeList(struct Node* head) {
100     struct Node* temp;
101     while (head != NULL) {
102         temp = head;
103         head = head->next;
104         free(temp);
105     }
106 }
107
108 int main(int argc, char const *argv[])
109 {
110
111
112
113
114
115     //Question 1
116     struct contact* addressbook=(struct contact*) malloc(30 * sizeof(struct contact) );
117     int cont_num=0;
118     while (1){
119         printf("choose:\n1.to insert a contact\n2.to delete a contact\n3.to exit.");
120         int choice;
121         scanf("%d",&choice);
122         if (choice == 1){
123             (cont_num)++;
124             addressbook=realloc(addressbook,100*sizeof(struct contact));
125             if (addressbook == NULL){
126                 printf("memory not allotted");
127                 break;
128             }
129             printf("enter name:");
130             scanf("%s",addressbook->name);
131             printf("enter email:");
132             scanf("%s",addressbook->email);

```

```

133         printf("enter phone number:");
134         scanf("%s",addressbook->numbr);
135         printf("contact saved successfully");
136
137     }
138     else if (choice == 2) {
139         char delnum[50];
140         printf("enter phone number to be deleted:");
141         scanf("%s",delnum);
142         for (int i=0;i<=cont_num;i++){
143             if (addressbook->numbr==delnum){
144                 for (int j=i;j<=cont_num;j++){
145                     strcpy((addressbook)[j].name, (addressbook)[j + 1].name);
146                     strcpy((addressbook)[j].email, (addressbook)[j + 1].email);
147                     strcpy((addressbook)[j].numbr, (addressbook)[j + 1].numbr);
148                 }
149             }
150         }
151
152     }
153
154 }
155 else if (choice == 3){
156     break;
157 }
158 }
159
160
161 free(addressbook);
162
163
164
165
166 //Question 2
167 struct Node* list1 = NULL;
168 struct Node* list2 = NULL;
169
170 insertEnd(&list1, 1);
171 insertEnd(&list1, 3);
172 insertEnd(&list1, 5);
173
174 insertEnd(&list2, 2);
175 insertEnd(&list2, 4);
176 insertEnd(&list2, 6);
177
178 printf("List 1: ");
179 printList(list1);
180
181 printf("List 2: ");
182 printList(list2);
183
184 struct Node* mergedList = mergeSortedLists(list1, list2);
185
186
187 printf("Merged List: ");
188 printList(mergedList);
189
190
191 freeList(list1);
192 freeList(list2);
193 freeList(mergedList);
194 return 0;
195 }
196
197
198

```

```

199
200 //Question 3
201 int count = 1;
202 struct linkedlist* head = (struct linkedlist*)malloc(sizeof(struct linkedlist));
203 head->next = NULL;
204
205 while (1) {
206     int check = 0;
207     printf("Enter 1 to enter data in linked list or 0 to exit: ");
208     scanf("%d", &check);
209
210     if (check == 1) {
211         struct linkedlist* node = (struct linkedlist*)malloc(sizeof(struct linkedlist));
212         printf("Enter number data for linked list: ");
213         scanf("%d", &node->data);
214
215         node->next = head->next;
216         head->next = node;
217         count++;
218     } else {
219         break;
220     }
221 }
222 int* array = (int*)malloc(count * sizeof(int));
223
224 struct linkedlist* current = head->next;
225 for (int i = 0; i < count; i++) {
226     array[i] = current->data;
227     current = current->next;
228 }
229
230 for (int i = 0; i < count; i++) {
231     printf("%d ", array[i]);
232 }
233
234
235
236
237 //Question 4
238
239 struct linkedlist* odd = (struct linkedlist*)malloc(sizeof(struct linkedlist));
240 odd->next = NULL;
241 struct linkedlist* curr = odd;
242
243 for (int j = 0; j <= 50; j++) {
244     curr->data = j;
245     curr->next = (struct linkedlist*)malloc(sizeof(struct linkedlist));
246     curr = curr->next;
247 }
248
249 curr->next = NULL;
250
251 curr = odd;
252 struct linkedlist* temp;
253 while (curr->next != NULL && curr->next->next != NULL) {
254     temp = curr->next;
255     curr->next = curr->next->next;
256     free(temp);
257     curr = curr->next;
258 }
259 curr = odd;
260 while (curr->next != NULL) {
261     printf("%d ", curr->data);
262     curr = curr->next;
263 }
264

```

