

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include "stdio.h"
3  #include "stdlib.h"
4  #include "malloc.h"
5  #include "string.h"
6
7  typedef struct Telefon
8  {
9      float pret;
10     char* marca;
11 } Telefon;
12
13 Telefon creareTelefon(float p, const char* m)
14 {
15     Telefon t;
16     t.pret = p;
17     t.marca = (char*) malloc(sizeof(char) * (strlen(m) + 1));
18     strcpy(t.marca, m);
19     return t;
20 }
21
22 void afisareTelefon(Telefon t)
23 {
24     printf("\n Telefonul cu marca %s are pretul %5.2f", t.marca, t.pret);
25 }
26
27 typedef struct Node
28 {
29     Telefon info;
30     struct Node* next;
31 } Node;
32
33 Node* creareNode(Node* next, Telefon t)
34 {
35     Node* nou = (Node*) malloc(sizeof(Node));
36     nou->info = creareTelefon(t.pret, t.marca);
37     nou->next = next;
38     return nou;
39 }
40
41 void afisareNode(Node* head)
42 {
43     if (head)
44     {
45         Node* aux = head;
46         while (aux) {
47             afisareTelefon(aux->info);
48             aux = aux->next;
49         }
50     }
```

```
50     }
51 }
52
53
54 //STACK
55
56 //inserare inceput
57 Node* pushStack(Node* head, Telefon t)
58 {
59     Node* nou = createNode(NULL, t);
60     if (head)
61     {
62         nou->next = head;
63         head = nou;
64     }
65     else {
66         head = nou;
67     }
68
69     return head;
70 }
71
72 //extragere inceput
73
74 Telefon popStack(Node** head)
75 {
76     if (*head) {
77         Node* aux = *head;
78         Telefon rezultat = createTelefon(aux->info.pret, aux->info.marca);
79         *head = (*head)->next; //se modifica direct in main, in timp real
80         free(aux->info.marca);
81         free(aux);
82         return rezultat;
83     }
84     else {
85         //returnam info redundanta
86         return createTelefon(-1, "");
87     }
88 }
89
90
91 //inserare final
92 Node* pushStack2(Node* head, Telefon t)
93 {
94     Node* nou = createNode(NULL, t);
95     if (head) {
96         Node* aux = head;
97         while (aux->next) {
98             aux = aux->next;
```

```
109     }
110     aux->next = nou;
111 }
112 else {
113     head = nou;
114 }
115 return head;
116 }
117
118 //extragere final
119 Telefon popStack2(Node** head)
120 {
121     if (*head) {
122         Node* aux = *head;
123         //stergem ultimul nod
124         //ne oprim pe penultimul
125         while (aux->next->next) {
126             aux = aux->next;
127         }
128         Telefon rezultate = creareTelefon(aux->next->info.pret, aux->next-
129             >info.marca);
130         free(aux->next->info.marca);
131         free(aux->next);
132         aux->next = NULL;
133         return rezultate;
134     }
135     else {
136         return creareTelefon(-1, "");
137     }
138 }
139
140 //QUEUE
141
142 //inserare final
143 Node* pushQueue(Node* head, Telefon t)
144 {
145     return pushStack2(head, t);
146 }
147
148 //extragere inceput
149 Telefon popQueue(Node** head) {
150     return popStack(head);
151 }
152
153 //inserare inceput
154 Node* pushQueue2(Node* head, Telefon t) {
155     return pushStack(head, t);
156 }
```

```
147
148 //extragere final
149 Telefon popQueue2(Node** head) {
150     return popStack2(head);
151 }
152
153 //HW 1 - inserare inainte unui nod cu un anumit pret
154 Node* inserareInainteaNodului(Node* head, Telefon t, float price)
155 {
156     Node* nou = createNode(NULL, t); //creat nod nou cu telefonul primit ca
    parametru
157     if (head) { //daca exista stiva
158         if (head->info.pret == price) {
159             nou->next = head;
160             head = nou;
161         }
162         else {
163             Node* p = head; //ne luam un auxiliar
164             while (p->next && p->next->info.pret != price) { //cat timp
    urmatorul nod are pretul diferit de cel
    //primit ca parametru, ma mut pe urmatorul nod
165                 p = p->next;
166             }
167             if (p->next->info.pret == price) // verificam din nou conditia
168             {
169                 //refacem legaturile
170                 nou->next = p->next;
171                 p->next = nou;
172             }
173         }
174     }
175     else {
176         head = nou;
177     }
178     return head;
179 }
180
181 //HW 2 - inserare dupa un nod cu un anumit pret
182 Node* inserareDupaUnNod(Node* head, Telefon t, float price)
183 {
184     Node* nou = createNode(NULL, t);
185     if (head) {
186         Node* p = head;
187         while (p->next && p->info.pret != price) {
188             p = p->next;
189         }
190         if (p->info.pret == price) {
191             nou->next = p->next;
192             p->next = nou;
193         }
```

```
194     }
195 }
196 return head;
197 }
198
199 //HW 3 - extragere nod cu un anumit pret
200 Telefon extragereNod(Node** head, float price)
201 {
202     if (*head) {
203         Node* p = *head;
204         if ((*head)->info.pret == price) {
205             Telefon rezultat = creareTelefon((*head)->info.pret, (*head)->info.marca);
206             Node* aux = (*head);
207             (*head) = (*head)->next;
208             free(aux->info.marca);
209             free(aux);
210             return rezultat;
211         }
212         else {
213
214             while (p->next && p->next->info.pret != price) {
215                 p = p->next;
216             }
217             if (p->next) {
218                 Telefon rezultat = creareTelefon(p->next->info.pret, p->next->info.marca);
219                 Node* aux = p->next;
220                 p->next = p->next->next;
221                 free(aux->next->info.marca);
222                 free(aux);
223
224                 return rezultat;
225             }
226             else {
227                 return creareTelefon(-1, "");
228             }
229         }
230     }
231 }
232 }
233 else {
234     return creareTelefon(-1, "");
235 }
236
237 }
238
239 //HW 4 - stack 2 queue
240 Node* Stack2Queue(Node* head)
```

```
241 {
242     Node* queue = NULL;
243     while (head) {
244         //1. scot din stack
245         //Telefon tmp = popStack(&head);
246
247         //2. pus in queue
248         //queue = pushQueue(queue, tmp);
249
250         //same thing but in one line
251         queue = pushQueue(queue, popStack(&head));
252     }
253 }
254
255 //HW 5 - Queue to Stack
256
257 Node* Queue2Stack(Node* head)
258 {
259     Node* stack = NULL;
260     while (head)
261     {
262         stack = pushStack(stack, popQueue(&head));
263     }
264 }
265 void main() {
266     Node* stack = NULL;
267     stack = pushStack(stack, createTelefon(6500, "Apple"));
268     stack = pushStack(stack, createTelefon(4500, "Samsung"));
269     stack = pushStack(stack, createTelefon(4200, "Huawei"));
270     stack = pushStack(stack, createTelefon(1500, "Nokia"));
271     stack = pushStack(stack, createTelefon(8700, "Apple"));
272
273     afisareNode(stack);
274
275
276     Telefon t = popStack(&stack);
277     printf("\n Afisare element extras ");
278     afisareTelefon(t);
279     //afisareTelefon(popStack(&stack));
280
281     printf("\n Afisare lista dupa extragere ");
282     afisareNode(stack);
283
284     printf("\n Afisare element extras ");
285     //afisareTelefon(t);
286     afisareTelefon(popStack2(&stack));
287
288     printf("\n Afisare lista dupa extragere ");
289     afisareNode(stack);
```

```
290
291     Telefon t1 = creareTelefon(100000, "GGHDDFFDHHVHD");
292     printf("\n afisare dupa inserare inainte");
293     stack = inserareInainteaNodului(stack, t1, 4200);
294     afisareNode(stack);
295
296
297
298     Telefon t2 = creareTelefon(5869, "GIGI");
299     printf("\n afisare dupa inserare dupa un nod");
300     stack = inserareDupaUnNod(stack, t2, 4500);
301     afisareNode(stack);
302
303
304     printf("\n Afisare element extras ");
305     //afisareTelefon(t);
306     afisareTelefon(extragereNod(&stack, 1500));
307     printf("\n Afisare lista dupa extragere mijloc");
308     afisareNode(stack);
309
310     printf("\n Stack -> queue");
311     Node* queue = Stack2Queue(stack);
312     afisareNode(queue);
313
314     printf("\n Queue -> Stack");
315     Node* stack2 = Queue2Stack(queue);
316     afisareNode(stack2);
317
318 }
319
320
```