

Structures

● Graded

Student

SRUTHI SUBRAMANIAN

Total Points

200 / 200 pts

Autograder Score

200.0 / 200.0

Passed Tests

Test 1 (10/10)

Test 10 (20/20)

Test 11 (20/20)

Test 12 (20/20)

Test 13 (20/20)

Test 14 (20/20)

Test 2 (10/10)

Test 3 (10/10)

Test 4 (10/10)

Test 5 (10/10)

Test 6 (10/10)

Test 7 (10/10)

Test 8 (10/10)

Test 9 (20/20)

Autograder Results

Test 1 (10/10)

Test 10 (20/20)

Test 11 (20/20)

Test 12 (20/20)

Test 13 (20/20)

Test 14 (20/20)

Test 2 (10/10)

Test 3 (10/10)

Test 4 (10/10)

Test 5 (10/10)

Test 6 (10/10)

Test 7 (10/10)

Test 8 (10/10)

Test 9 (20/20)

Submitted Files

```
1  #include <stdio.h>
2
3  int main() {
4      // Write C code here
5      int o, MAX;
6      MAX = 0;
7      scanf("%d %d", &o, &MAX);
8      if (o == 0) {
9          //stack
10         int a, b, i;
11         int top = 0;
12         //top is 1 index above the highest element: when it is MAX, the stack is full
13         a = -1;
14         while (a != 3) {
15             scanf("%d", &a);
16             int stack[MAX + 2];
17             if (a == 0) {
18                 //print the stack - bottom to top
19                 if (top > 0) {
20                     for (i = 0; i < top; i++) {
21                         printf("%d ", stack[i]);
22                     }
23                 }
24                 printf("\n");
25             }
26             if (a == 1) {
27                 scanf("%d", &b);
28                 stack[top] = b;
29                 //push b to the stack
30                 if (top == MAX) {
31                     //stack is full
32                     printf("-1\n");
33                     break;
34                 }
35             }
36             else {
37                 top = top + 1;
38             }
39             if (a == 2) {
40                 //pop
41                 if (top == 0) {
42                     printf("-1\n");
43                     break;
44                 }
45                 top = top - 1;
46                 printf("%d\n", stack[top]);
47                 stack[top] = 0;
48             }
49         }
```

```

50 }
51 else if (o == 1) {
52     int queue[MAX];
53     int a, b, i, head, tail;
54     head = 0;
55     tail = 0;
56     b = 0;
57     while (a != 3) {
58         scanf("%d ", &a);
59         if (a == 0) {
60             //print the queue
61             if (tail != head) {
62                 //the queue is not empty
63                 if (head < tail) {
64                     for (i = head; i < tail; i++) {
65                         printf("%d ", queue[i]);
66                     }
67                 }
68                 else {
69                     for (i = head; i < MAX; i++) {
70                         printf("%d ", queue[i]);
71                     }
72                     for (i = 0; i < tail; i++) {
73                         printf("%d ", queue[i]);
74                     }
75                 }
76             }
77             printf("\n");
78         }
79         else if (a == 1) {
80             //enqueue b
81             scanf("%d\n", &b);
82             if (head == tail) {
83                 queue[head] = b;
84                 tail = tail + 1;
85             }
86             else if (head < tail) {
87                 if (head == 0) {
88                     if (tail == MAX - 1) {
89                         printf("-1\n");
90                         break;
91                     }
92                     else {
93                         tail = tail + 1;
94                         queue[tail - 1] = b;
95                     }
96                 }
97                 else {
98                     if (tail == MAX - 1) {
99                         tail = 0;
100                         queue[MAX - 1] = b;
101                     }

```

```

102         else {
103             tail = tail + 1;
104             queue[tail - 1] = b;
105         }
106     }
107 }
108 else {
109     //head>=tail
110     if (tail + 1 == head) {
111         printf("-1\n");
112         break;
113     }
114     else {
115         tail = tail + 1;
116         queue[tail - 1] = b;
117     }
118 }
119 }
120 if (a == 2) {
121     //dequeue - first check if empty
122     if (head == tail) {
123         printf("-1\n");
124         break;
125     }
126     else if (head == MAX - 1) {
127         int temp;
128         temp = queue[head];
129         head = 0;
130         printf("%d\n", temp);
131     }
132     else {
133         head = head + 1;
134         printf("%d\n", queue[head - 1]);
135     }
136 }
137 }
138 }
139 else if (o == 2) {
140     int a, i;
141     int* heap;
142     a = -1;
143     int size; //the size of the heap array
144     size = 0;
145     int heapp[MAX];
146     heap = heapp;
147     int* MinHeapify(int* heap, int ind) {
148         //1 indexing!!
149         int minind = ind;
150         if (ind * 2 <= size && heap[2 * ind - 1] < heap[ind - 1]) {
151             minind = 2 * ind;
152         }
153         if (ind * 2 + 1 <= size && heap[2 * ind] < heap[minind - 1]) {

```

```

154     minind = 2 * ind + 1;
155 }
156 if (minind != ind) {
157     //swap minind term and X and then call MinHeapify on minind
158     int X = ind;
159     int t = heap[X - 1];
160     heap[X - 1] = heap[minind - 1];
161     heap[minind - 1] = t;
162     return MinHeapify(heap, minind);
163 }
164 else {
165     return heap;
166 }
167 }
168 while (a != 4) {
169     scanf("%d ", &a);
170     if (a == 0) {
171         //print the heap in the usual order of the array
172         for (i = 0; i < size; i++) {
173             printf("%d ", heap[i]);
174         }
175         printf("\n");
176     }
177     else if (a == 1) {
178         //build-heap
179         int a2;
180         int b;
181         scanf("%d ", &a2);
182         size = a2;
183         if (a2 > MAX) {
184             printf("-1\n");
185             break;
186         }
187         for (i = 0; i < a2; i++) {
188             scanf("%d ", &b);
189             heap[i] = b;
190         }
191         for (i = size; i > 0; i--) {
192             heap = MinHeapify(heap, i);
193         }
194     }
195     else if (a == 2) {
196         //decrease-key
197         int a2, a3;
198         scanf("%d %d", &a2, &a3);
199         if (a2 <= size) {
200             if (heap[a2 - 1] > a3) {
201                 heap[a2 - 1] = a3;
202                 for (i = size; i > 0; i--) {
203                     heap = MinHeapify(heap, i);
204                 }
205             }

```

```
206     }
207 }
208 else if (a == 3) {
209     //extract minimum
210     if (size == 0) {
211         printf("-1\n");
212         break;
213     }
214     int min = heap[0];
215     size = size - 1;
216     heap[0] = heap[size];
217     heap[size] = min;
218     heap = MinHeapify(heap, 1);
219     printf("%d\n", min);
220 }
221 }
222 }
223
224
225 return 0;
226 }
```
