

## **\*문제\***

배열을 이용하여 **이진 탐색 트리 클래스**를 구현하십시오. 이 클래스는 다음과 같은 기능을 포함하고 있습니다.

1. 양수가 입력되면 이진 탐색 트리에 삽입 합니다.
2. 음수가 입력되면 그 수의 절대값을 탐색하여 해당 값을 출력하고 트리에서 삭제합니다.  
이 때, 만약 해당 값이 트리에 없다면 0을 출력합니다. 단, 배열의 값을 모두 출력할 때 0은 생략합니다.
3. 0이 입력되면 배열의 전체 값을 출력하고 프로그램을 종료합니다.

### **입력 형식 :**

1. 동시에 트리에 들어가는 숫자는 8개 이하입니다.
2. 중복은 존재하지 않기 때문에 별도의 처리는 불필요 합니다.
3. 배열의 0 번째 자리는 사용하지 않습니다. 또한, 배열의 값을 출력할 때에도 0 번째 자리는 생략합니다.

### **힌트:**

1. search, insert, delete 함수가 필요합니다.
2. 배열 구현이므로 delete함수 구현 시 재귀함수 호출이 필요할 수 있습니다. 따라서 인덱스를 받으면 해당 인덱스를 삭제해주는 deleter 함수를 별도로 구현하는 것이 용이할 수 있습니다.
3. 특정한 값의 인덱스를 찾는 getIndex 함수를 만들면 편할 수 있습니다.
4. 출력 후 삭제를 구현할 때에는 메인 함수에서 search와 delete 함수를 연달아 호출하는 것이 간단합니다.
5. 삭제를 구현할 때 다양한 방법이 있습니다. 어떤 방법이든 채점이 가능하도록 테스트 케이스가 구성되어 있으므로 자유롭게 구현하면 됩니다.

### **제한 사항:**

1. 클래스로 구현합니다(C 언어의 경우 struct 사용)

### 입출력 예시 1:

```
8
1
7
2
6
3
5
4
-7
7
-7
0
-6
6
-5
5
-4
4
-3
3
-2
2
-1
1
0
8
```

### 예시 힌트 1

0을 입력하기 전에는 프로그램이 종료되지 않기 때문에 음수 값을 입력하면 절대값을 취한 후 해당 값을 찾아서 출력 및 삭제가 일어납니다. 마지막에 0이 입력되면 남아있는 값을 출력한 후 종료합니다.

### 입력 예시 2:

```
50
100
1
99
2
98
3
97
0
```

### 출력 예시 2:

```
50 1 100 2 99 3 98 97
```

### 예시 힌트 2

입력된 값을 토대로 BST를 그려보는 것을 추천합니다. 출력 예시는 root 노드부터 차례대로 depth를 기반으로 출력(배열로 구현했기 때문에) 된 모습입니다.

C / C++ 를 사용하시는 학생 분들은 아래의 품을 참고해서 작성해 주셔야 기본적인 컴파일 에러를 방지할 수 있습니다.

또한 C 언어의 경우 표준 컴파일러에서는 scanf\_s 또는 printf\_s 등과 같이 "\_s"를 붙이는 경우 컴파일 에러가 발생하기 때문에 "\_s"를 제거한 scanf / printf 등의 함수를 사용하시기 바랍니다.

## C:

```
#include <stdio.h>

int main() {
    /* TODO */

    return 0;
}
```

## C++:

```
#include <iostream>
using namespace std;

int main() {
    /* TODO */

    return 0;
}
```

## 모범답안

```
1  #include <iostream>
2  #include <cstdlib>
3  using namespace std;
4
5  class ctree {
6      int tree[500] = { 0, };
7      int size = 0;
8      void puller(int, int);
9
10 public:
11     int getidx(int idx) { return tree[idx]; }
12     void insert(int, int);
13     int search(int, int);
14     void tdelete(int);
15     int findMax(int);
16     int findMin(int);
17 };
18
19 void ctree::insert(int v, int idx) {
20     if (size == 0) {
21         tree[1] = v;
22         size = 1;
23         return;
24     }
25     if (v < tree[idx]) {
26         if (tree[2 * idx] == 0) { tree[idx * 2] = v; }
27         else insert(v, 2 * idx);
28     }
29     else {
30         if (tree[2 * idx + 1] == 0) { tree[idx * 2 + 1] = v; }
31         else insert(v, 2 * idx + 1);
32     }
33 }
34
35 int ctree::search(int v, int idx) {
36     int result;
37
38     if (tree[idx] == 0) return -1;
39     else if (v < tree[idx]) {
40         result = search(v, 2 * idx);
41     }
42     else if (v > tree[idx]) {
43         result = search(v, 2 * idx + 1);
44     }
45     else {
46         return idx;
47     }
48     return result;
49 }
50 }
```

```

52 ▾ int ctree::findMax(int idx) {
53 ▾     if (tree[2 * idx + 1] != 0) {
54         return findMax(2 * idx + 1);
55     }
56     else return idx;
57 }
58
59 ▾ int ctree::findMin(int idx) {
60 ▾     if (tree[2 * idx] != 0) {
61         return findMin(2 * idx);
62     }
63     else return idx;
64 }
65
66
67 ▾ void ctree::puller(int temp, int idx) {
68
69 ▾     if (tree[temp * 2] == 0 && tree[temp * 2 + 1] == 0) {
70         tree[temp] = 0;
71         return;
72     }
73
74     tree[idx * 2] = tree[temp * 2];
75     tree[temp * 2] = 0;
76     tree[idx * 2 + 1] = tree[temp * 2 + 1];
77     tree[temp * 2 + 1] = 0;
78
79     puller(temp * 2, idx * 2);
80     puller(temp * 2 + 1, idx * 2 + 1);
81
82
83 }
84
85 void ctree::tdelete(int idx)
86 ▾ {
87 ▾     if ((tree[idx * 2] == 0) && (tree[idx * 2 + 1] == 0)) {
88         tree[idx] = 0;
89         return;
90     }
91
92 ▾     if ((tree[idx * 2] != 0) && (tree[idx * 2 + 1] == 0)) {
93         tree[idx] = tree[idx * 2];
94         puller(idx * 2, idx);
95     }
96 ▾     else if (tree[idx * 2] == 0 && (tree[idx * 2 + 1] != 0)) {
97         tree[idx] = tree[idx * 2 + 1];
98         puller(idx * 2 + 1, idx);
99     }
100 ▾     else {
101         int i = idx * 2;
102         tree[i] = tree[i * 2 + 1];
103         tree[idx] = tree[i / 2];
104         tdelete(i / 2);
105     }
106 }

```

```

108 ▾ int main() {
109     int input = 1;
110     int delidx;
111     ctree tree = ctree();
112 ▾ while (input != 0) {
113         cin >> input;
114         if (input > 0) tree.insert(input, 1);
115 ▾     else if (input < 0) {
116         input = abs(input);
117         delidx = tree.search(input, 1);
118 ▾         if (delidx == -1) {
119             cout << 0 << endl;
120         }
121 ▾     else {
122         cout << tree.getidx(delidx) << endl;
123         tree.tdelete(delidx);
124     }
125 }
126 ▾ else {
127 ▾     for (int i = 1; i <= 299; i++) {
128         if (tree.getidx(i) == 0) continue;
129         else cout << tree.getidx(i) << " ";
130     }
131 }
132 }
133 }
134 }
135 }

```