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

void swap(int arr[], int n, int i)
{
    int temp = arr[n];
    arr[n] = arr[i];
    arr[i] = temp;
}

void heapify(int arr[], int n, int i)
{
    int l_c = 2 * i + 1;
    int r_c = 2 * i + 2;
    int large = i;

    if(l_c < n && arr[l_c] > arr[large])
    {
        large = l_c;
    }
    if(r_c < n && arr[r_c] > arr[large])
    {
        large = r_c;
    }

    if(large != i)
    {
        swap(arr, i, large);
        heapify(arr, n, large);
    }
}

void build_heap(int arr[], int n)
{
    for(int i = (n - 1) / 2; i >= 0; i--)
    {
        heapify(arr, n, i);
    }
}

int height(int arr[], int n)
{
    int i = 0;
    int height = 0;

```

```
47:     while(i < n)
48:     {
49:         height++;
50:         i = 2 * i + 1;
51:     }
52:     return height;
53: }
54:
55: void heapTree(int arr[], int n)
56: {
57:     int h = height(arr, n);
58:     int blocks = pow(2, h);
59:     int temp_blocks = blocks;
60:     int index = 0;
61:     int printed_index;
62:     int k = 3;
63:
64:     for(int i = 0; i < h; i++)
65:     {
66:         for(int j = 0; j < blocks; j++)
67:         {
68:             if(j == temp_blocks / 2 && index < n)
69:             {
70:                 cout << arr[index];
71:                 index++;
72:                 printed_index = j;
73:             }
74:             else
75:             {
76:                 if(j == (temp_blocks * k) / 2 && index < n)
77:                 {
78:                     cout << arr[index];
79:                     index++;
80:                     k += 2;
81:                 }
82:                 else
83:                 {
84:                     cout << " ";
85:                 }
86:             }
87:         }
88:         cout << "\n";
89:         printed_index = 0;
90:         temp_blocks = temp_blocks / 2;
91:         k = 3;
92:     }
```

```
 93: }
 94:
 95: void deleteElt(int arr[], int& n, int i)
 96: {
 97:     int last_elt = arr[n - 1];
 98:
 99:     arr[i] = last_elt;
100:
101:     n -= 1;
102:
103:     build_heap(arr, n);
104: }
105:
106: void insertElt(int arr[], int& n, int i)
107: {
108:     n += 1;
109:
110:     arr[n - 1] = i;
111:
112:     build_heap(arr, n);
113: }
114:
115: int main()
116: {
117:     int n;
118:     cout << "Enter the length:\n";
119:     cin >> n;
120:
121:     int arr[n];
122:     cout << "Enter elements: \n";
123:     for(int i = 0; i < n; i++)
124:     {
125:         cin >> arr[i];
126:     }
127:
128:     build_heap(arr, n);
129:     heapTree(arr, n);
130:
131:     //Deletion
132:     int i;
133:
134:     cout << "By looking at the tree enter ";
135:     cout << "the index of your element you";
136:     cout << " want to delete: \n";
137:
138:     cin >> i;
```

```
139:     cout << "After deleting '" << arr[i];
140:     cout << "' New heap tree is: \n";
141:     deleteElt(arr, n, i);
142:     heapTree(arr, n);
143:
144:
145:     //Insertion
146:     int j;
147:
148:     cout << "Enter the element you want to insert:\n";
149:     cin >> j;
150:
151:     insertElt(arr, n, j);
152:     cout << "New heap tree is: \n";
153:     heapTree(arr, n);
154:
155:
156:     return 0;
157: }
```