```c
// Max-Heap data structure in C

#include <stdio.h>
int size = 0;
void swap(int *a, int *b)
{
  int temp = *b;
  *b = *a;
  *a = temp;
}
void heapify(int array[], int size, int i)
{
  if (size == 1)
  {
    printf("Single element in the heap");
  }
  else
  {
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;
    if (l < size && array[l] > array[largest])
      largest = l;
    if (r < size && array[r] > array[largest])
      largest = r;
    if (largest != i)
    {
      swap(&array[i], &array[largest]);
      heapify(array, size, largest);
    }
  }
}
void insert(int array[], int newNum)
{
  if (size == 0)
  {
    array[0] = newNum;
    size += 1;
  }
```

```
40:   else
41:   {
42:     array[size] = newNum;
43:     size += 1;
44:     for (int i = size / 2 - 1; i >= 0; i--)
45:     {
46:       heapify(array, size, i);
47:     }
48:   }
49: }
50: void deleteRoot(int array[], int num)
51: {
52:   int i;
53:   for (i = 0; i < size; i++)
54:   {
55:     if (num == array[i])
56:       break;
57:   }
58:
59:   swap(&array[i], &array[size - 1]);
60:   size -= 1;
61:   for (int i = size / 2 - 1; i >= 0; i--)
62:   {
63:     heapify(array, size, i);
64:   }
65: }
66: void printArray(int array[], int size)
67: {
68:   for (int i = 0; i < size; ++i)
69:     printf("%d ", array[i]);
70:   printf("\n");
71: }
72: int main()
73: {
74:   int array[10];
75:
76:   insert(array, 3);
77:   insert(array, 4);
78:   insert(array, 9);
```

```
79:   insert(array, 5);
80:   insert(array, 2);
81:
82:   printf("Max-Heap array: ");
83:   printArray(array, size);
84:
85:   deleteRoot(array, 4);
86:
87:   printf("After deleting an element: ");
88:
89:   printArray(array, size);
90: }
```