Delete Arbitarty element in a min-heap

```
#include <stdio.h>
#include <stdlib.h>

void swap(int *a, int *b)
{
        int temp = *a;
        *a = *b;
        *b = temp;
}

void MinHeapify(int *arr, int index, int size)
{
        int left = 2*index + 1;
        int right = 2*index + 2;
        int smallest = index;

        if (left < size && arr[left] < arr[index])
      smallest = left;
   if (right < size && arr[right] < arr[smallest])
      smallest = right;
   if (smallest != index)
   {
      swap(&arr[index], &arr[smallest]);
      MinHeapify(arr, index, smallest);
   }
}

void buildMinHeap(int *arr, int size)
{
        for(int index = size/2 -1; index >= 0; index-- )
                MinHeapify(arr, index, size);
}

void deleteEleFromMinHeap(int *arr, int *size, int ele)
{
        int index;
        //search index of elemet

        for(index = 0; index < *size; index++)
                if(arr[index] == ele)
                        break;

        //element found
        if(index < *size)
        {
                arr[index] = arr[*size - 1];
                *size = *size - 1;
                MinHeapify(arr, index, *size);
        }
}
```

```
void printMinHeap(int *arr, int size)
{
	for(int index = 0; index < size; index++)
		printf("%d\t", arr[index]);
}

int main()
{
	int *arr, size, ele;
	printf("Enter size of heap\n");
	scanf("%d", &size);

	//allocate memory
	arr = (int *)malloc(sizeof(int) * size);

	printf("Enter elements in heap\n");
	for(int index = 0; index < size; index++)
		scanf("%d", &arr[index]);

	buildMinHeap(arr, size);
	printf("Enter element to delete\n");
	scanf("%d", &ele);

	deleteEleFromMinHeap(arr, &size, ele);
	printMinHeap(arr, size);

}
```

Time complexity: O(n)
Space complexity: O(1)