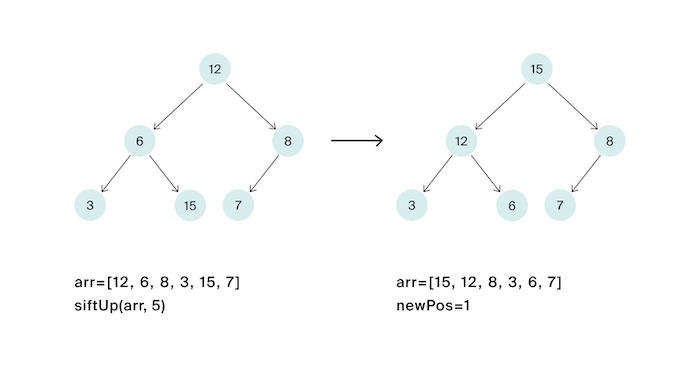
M. Просеивание вверх

Напишите функцию, совершающую просеивание вверх в куче на максимум. Гарантируется, что порядок элементов в куче может быть нарушен только элементом, от которого запускается просеивание.

Функция принимает в качестве аргументов массив, в котором хранятся элементы кучи, и индекс элемента, от которого надо сделать просеивание вверх. Функция должна вернуть индекс, на котором элемент оказался после просеивания. Также необходимо изменить порядок элементов в переданном в функцию массиве.

Индексация в массиве, содержащем кучу, начинается с единицы. Таким образом, сыновья вершины на позиции v это 2v и 2v+1. Обратите внимание, что нулевой элемент в передаваемом массиве фиктивный, вершина кучи соответствует 1-му элементу.



Формат ввода

Элементы кучи —– целые числа, лежащие в диапазоне от −109 до 109. Все элементы кучи уникальны. Передаваемый в функцию индекс лежит в диапазоне от 1 до размера передаваемого массива. В куче содержится от 1 до 105 элементов.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/XtYa7EqfPz6bmA).

**Python**

**def** sift\_up(heap: **list**, idx: **int**) -> **int**

**C++**

**#include** "solution.h" *// Attention!*   
**int** siftUp(std::vector<**int**>& heap, **int** idx);

**Java**

**public** **class** Solution {   
    **public** **static** **int** siftUp(**int**[] heap, **int** idx) {   
*// your code*   
    }   
}

**GO**

func siftUp(heap []**int**, idx **int**) **int**;

**NodeJS**  
Пожалуйста, не используйте стрелочную нотацию при определении функции

function siftUp(heap, idx) {   
*// Your code*   
}