https://leetcode.com/problems/last-stone-weight/submissions/

```
class Solution {
public:
   int lastStoneWeight(vector<int>& stones) {
       // 2 7 4 1 8 1
       // 8 7 4 2 1 1
       // 1 1 2 4 7 8
       // 1 1 1 2 4
       // 1 1 1 2
       // 1 1 1
       // 1
       int _size = stones.size();
       priority_queue<int> q;
       for(int i=0;i<_size;i++){
           q.push(stones[i]);
       }
       while(!q.empty() and q.size()>=2){
               int y = q.top();
               q.pop();
               int x = q.top();
               q.pop();
               if(x!=y)
                   q.push(y-x);
       }
       return q.size() == 0? 0: q.top();
   }
};
```

Implement a custom filtering function that takes an integer array arr and a filtering function fn and returns a new array containing elements where fn(arr[i], i) evaluates to a truthy value. Please note that you should not use the built-in Array.filter method to solve this lab.

Your task is to create a custom filtering function that filters elements in an array based on a given filtering function, fn. The function should take an array arr and a filtering function fn as arguments. It should return a new array where only the elements that satisfy the filtering function are included.

Make sure to consider the following points:

Your function should work for arrays of different lengths and with elements of different types.

It should handle truthy and falsy values correctly.

Here are some example cases to help you understand the problem better:

Example 1:

```
const arr = [0, 10, 20, 30];
const fn = function greaterThan10(n) {
  return n > 10;
}
const newArray = filter(arr, fn); // [20, 30]
```

In this example, the function filters out values that are not greater than 10. The returned array should only contain the values 20 and 30.

Example 2:

```
const arr = [1, 2, 3];
const fn = function firstIndex(n, i) {
  return i === 0;
}
const newArray = filter(arr, fn); // [1]
```

In the second example, fn also accepts the index of each element. The function removes elements not at index 0, so the returned array should only contain the value 1.

Example 3:

```
const arr = [-2, -1, 0, 1, 2];
const fn = function plusOne(n) {
  return n + 1;
}
const newArray = filter(arr, fn); // [-2, 0, 1, 2]
```

In the last example, the function filters out falsey values such as 0. The returned array should only include the values -2, 0, 1, and 2.

https://www.programiz.com/javascript/online-compiler/

```
// Online Javascript Editor for free
// Write, Edit and Run your Javascript code using JS Online Compiler

function filter(arr, fn){
    let len = arr.length;
    const newArray = [];
    for(let idx = 0;idx<len;idx++){
        if(fn(arr[idx], idx))
            newArray.push(arr[idx]);
    }
    return newArray;
}

// const arr = [0, 10, 20, 30];
// const fn = function greaterThan10(n, i=0) {
//   return n > 10;
// }
// const newArray = filter(arr, fn); // [20, 30]
// console.log(newArray)

// const arr = [1, 2, 3];
// const fn = function firstIndex(n, i) {
//   return i === 0;
// }
// const newArray = filter(arr, fn); // [1]
// console.log(newArray)


const arr = [-2, -1, 0, 1, 2];
const fn = function plusOne(n) {
  return n + 1;
}
const newArray = filter(arr, fn); // [-2, 0, 1, 2]
console.log(newArray)

console.log("Try programiz.pro");
```