Lab: Advanced Arrays

Problems for lab for the "PHP Fundamentals" course @ SoftUni.

You can check your solutions in Judge.

1. Remove Negatives and Reverse

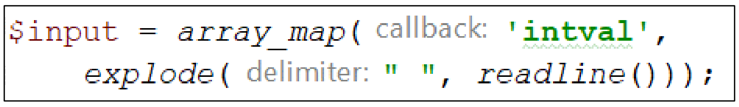
Read an **array of integers**, **remove all negative numbers** from it and print the remaining elements in **reversed order**. In case of no elements left in the array, print "**`empty`**".

Examples

Input	Output
`10 -5 7 9 -33 50`	`50 9 7 10`
`7 -2 -10 1`	`1 7`
`-1 -2 -3`	`empty`

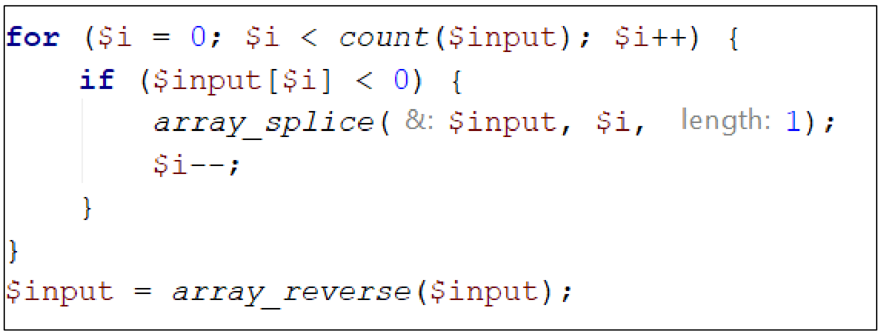
Solution

Read an array of integers.



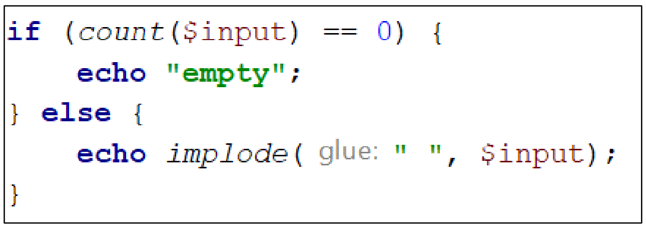
```
$input = array_map( callback: 'intval',
    explode( delimiter: " ", readline()));
```

Remove all negative numbers and reverse the collection.



```
for ($i = 0; $i < count($input); $i++) {
    if ($input[$i] < 0) {
        array_splice( &: $input, $i,  length: 1);
        $i--;
    }
}
$input = array_reverse($input);
```

If the array is empty print "empty", otherwise print all numbers joined by space.



```
if (count($input) == 0) {
    echo "empty";
} else {
    echo implode( glue: " ", $input);
}
```

2. Sum Adjacent Equal Numbers

Write a program to **sum all adjacent equal numbers** in a array of real numbers, starting from **left to right**.

- After two numbers are summed, the obtained result could be equal to some of its neighbors and should be summed as well (see the examples below).
- Always sum the **leftmost** two equal neighbors (if several couples of equal neighbors are available).

Examples

Input	Output	Explanation
3 3 6 1	12 1	**3 3** 6 1 → **6 6** 1 → 12 1
8 2 2 4 8 16	16 8 16	8 **2 2** 4 8 16 → 8 **4 4** 8 16 → **8 8** 8 16 → 16 8 16
5 4 2 1 1 4	5 8 4	5 4 2 **1 1** 4 → 5 4 **2 2** 4 → 5 **4 4** 4 → 5 8 4

Solution

Read an array from numbers.

```
$input = array_map( callback: 'floatval',
    explode( delimiter: ' ', readline()));
```

Iterate through the elements. Check if the number at the current index is equal to the next number. If it is, aggregate the numbers and reset the loop, otherwise don't do anything.

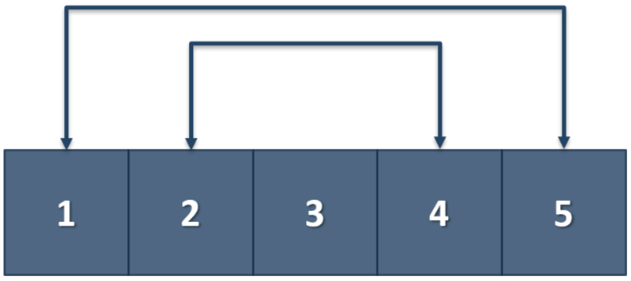
```
if ($input[$i] == $input[$i + 1]) {
    $element = $input[$i] + $input[$i + 1];
    $input[$i] = $element;
    array_splice( &: $input, offset: $i + 1, length: 1);
    $i = -1;
}
```

Finally, you have to print the numbers joined by space.

```
echo implode( glue: " ", $input);
```

3. Gauss' Trick

Write a program that **sum** all **numbers in an array** in the following order:
first **+** last, first + 1 **+** last - 1, first + 2 **+** last **-** 2**,** … first + n, last - n.



Examples

Input	Output
1 2 3 4 5	6 6 3
1 2 3 4	5 5

4. Array of products

Read a number **n** and **n lines of products**. Print a **numbered array** of all the products **ordered by name**.

Example

Input	Output
4 Potatoes Tomatoes Onions Apples	1.Apples 2.Onions 3.Potatoes 4.Tomatoes

Solution

First, we need to read the number **n** from the console

```php
$n = intval(readline());
```

Then we need to create our **array**

```php
$n = intval(readline());
$arr = [];
```

Then we need to iterate **n times** and **read products**.

```php
for ($i = 0; $i < $n; $i++) {
    $element = readline();
}
```

The next step is to add the current product to the array

```php
for ($i = 0; $i < $n; $i++) {
    $element = readline();
    $arr[$i] = $element;
}
```

After we finish reading the products we **sort our array alphabetically**

```php
 sort( &array: $arr);
```

The **sort method** sorts the array in ascending order.

Finally, we have to **print our sorted** array. To do that we **iterate through the array**.

```php
for ($i = 0; $i < count($arr); $i++) {
    echo $i + 1 . "." . $arr[$i] . PHP_EOL;
}
```

We use **i + 1**, because we want to **start counting from 1**.

5. Array Manipulation Basics

Write a program that reads an array of integers. Then until you receive **"end"**, you will be given different **commands:**

Add {number}: add a number to the end of the array

Remove {number}: remove number from the array

RemoveAt {index}: removes number at a given index

Insert {number} {index}: inserts a number at a given index

Note: All the indices will be valid!

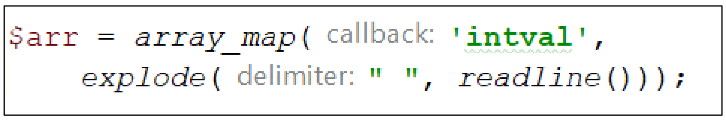
When you receive the **"end"** command print the **final state** of the array (**separated by spaces**)

Example

Input	Output
4 19 2 53 6 43 Add 3 Remove 2 RemoveAt 1 Insert 8 3 end	4 53 6 8 43 3

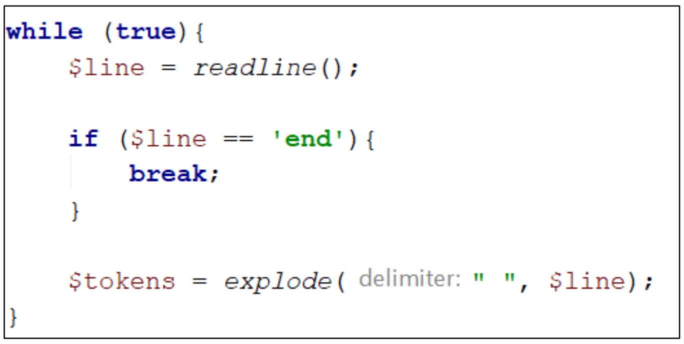
Solution

First let us read the array from the console.



```
$arr = array_map( callback: 'intval',
    explode( delimiter: " ", readline()));
```

Next we make the while loop for the commands and make switch statement for the commands

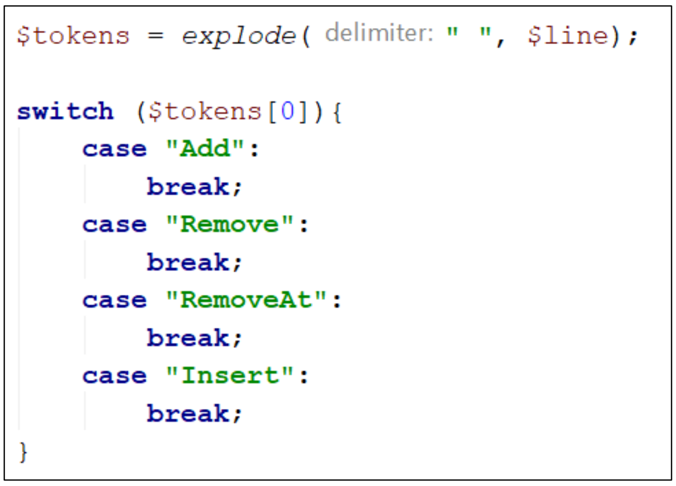


```
while (true){
    $line = readline();

    if ($line == 'end'){
        break;
    }

    $tokens = explode( delimiter: " ", $line);
}
```

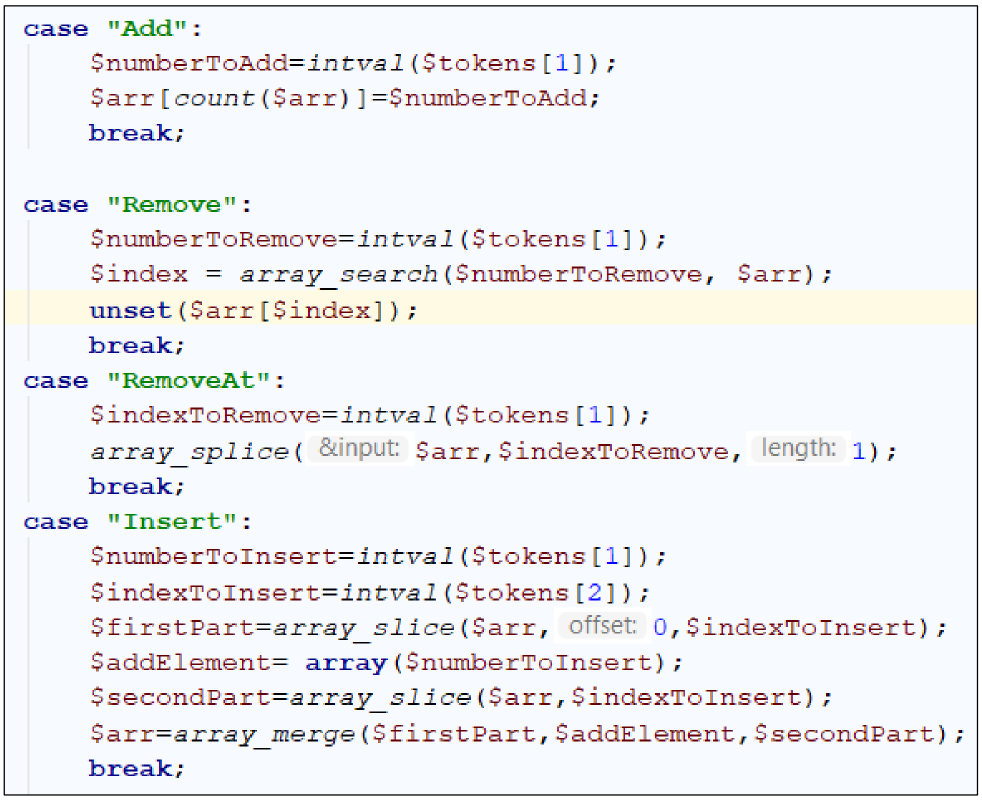
We break if the line is "end", otherwise we split it into tokens and process the command.



```php
$tokens = explode( delimiter: " ", $line);

switch ($tokens[0]){
    case "Add":
        break;
    case "Remove":
        break;
    case "RemoveAt":
        break;
    case "Insert":
        break;
}
```

Now let's implement each command.

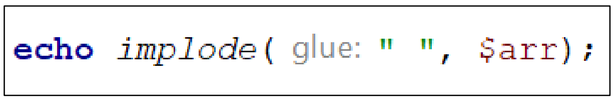


```php
case "Add":
    $numberToAdd=intval($tokens[1]);
    $arr[count($arr)]=$numberToAdd;
    break;

case "Remove":
    $numberToRemove=intval($tokens[1]);
    $index = array_search($numberToRemove, $arr);
    unset($arr[$index]);
    break;
case "RemoveAt":
    $indexToRemove=intval($tokens[1]);
    array_splice( &input: $arr,$indexToRemove, length: 1);
    break;
case "Insert":
    $numberToInsert=intval($tokens[1]);
    $indexToInsert=intval($tokens[2]);
    $firstPart=array_slice($arr, offset: 0,$indexToInsert);
    $addElement= array($numberToInsert);
    $secondPart=array_slice($arr,$indexToInsert);
    $arr=array_merge($firstPart,$addElement,$secondPart);
    break;
```

For all commands **except from** the **"Insert"**, **`tokens[1]`** is the **number/index**. For the **"Insert"** command we receive a **number and an index** (**`tokens[1], tokens[2]`**)

Finally, we **print** the numbers, joined by **a single space**



```php
echo implode( glue: " ", $arr);
```

6. Array Manipulation Advanced

Now we need to extend the functionality of the previous task and we will implement more complicated array commands. Again, read an array, and until you receive **"end"** read commands:

Contains {number} – check if the array contains the number. If **yes** print **"Yes"**, **otherwise** print **"No such number"**

Print even – print **all the numbers** that are **even separated by a space**

Print odd – print **all the numbers** that are **odd separated by a space**

Get sum – print the **sum of all the numbers**

Filter ({condition} {number}) – print all the numbers that **fulfill that condition**. The condition will be either **'<', '>', ">=", "<="**

Example

Input	Output

`2 13 43 876 342 23 543`
`Contains 100`
`Contains 543`
`Print even`
`Print odd`
`Get sum`
`Filter >= 43`
`Filter < 100`
`end`
	`No such number`
	`Yes`
	`2 876 342`
	`13 43 23 543`
	`1842`
	`43 876 342 543`
	`2 13 43 23`

7. Merging Arrays

You are going to receive two arrays with numbers. Create a result array which contains the numbers from both of the arrays. The first element should be from the first array, the second from the second array and so on. If the length of the two arrays are not equal, just add the remaining elements at the end of the array.

Examples

Input	Output

`3 5 2 43 12 3 54 10 23`
`76 5 34 2 4 12`
	`3 76 5 5 2 34 43 2 12 4 3 12 54 10 23`

`76 5 34 2 4 12`
`3 5 2 43 12 3 54 10 23`
	`76 3 5 5 34 2 2 43 4 12 12 3 54 10 23`

Hint

- Read the two arrays
- Create a result arrays
- Start looping through them until you reach the end of the smallest one
- Finally add the remaining elements (if any) to the end of the array