Implement a function that takes two lists of integers: `arr1` and `arr2`, subtracts one list from another and returns the result.

It should remove all values from `arr1`, which are present in list `arr2` keeping their order.

```
difference(List(1,2), List(1)) == List(2)
```

If a value is present in `arr2`, all of its occurrences must be removed from the other:

```
difference(List(1,2,2,2,3), List(2)) == List(1,3)
```

Given a list of a list of integers, determine whether each list is odd or even.

Write a function that outputs a list of strings wherein each string is either "odd" or "even" depending on whether the corresponding list in the argument has an odd sum or not.

If a list is empty consider it as `List(0)`.

Examples:

```
oddOrEven(List(
    List(1, 2, 3),
    List(2, -1, -5),
    List(2, 3)
)) -> List("even", "even", "odd")
```

Create a function that takes a list of non-negative integers and strings, then returns a new list with the strings removed.

Example:

```
removeStrings(List(1,2,"a","b")) == List(1,2)
removeStrings(List(1,"a","b",0,15)) == List(1,0,15)
removeStrings(List(1,2,"aasf","1","123",123)) == List(1,2,123)
```

Template:

```
def removeStrings(arr: List[Any]): List[Int] = {
    // your code here
}
```

Make a function that squares every digit of a number then concatenates them.

For example, if we run 7337 through the function, 499949 will be the result because 7 squared is 49 and 3 squared is 9.

Note: The function accepts an integer and returns an integer.

Implement a function that takes one input: `ceiling`, the function should return the sum of all positive multiples of 5 **or** 7 which are less than `ceiling`.

Examples:

```
sumify(10) --> 5+7=12
sumify(38) --> 210
```

Write a function that takes a list of Integers then removes all integers which are **not** perfect squares.

Example:

```
haveToBePerfect(List(25, 0, -5, 24, 1)) --> List(25, 0, 1)
```