# Day 21: Generics

**Objective**

Today we're discussing Generics; be aware that *not all languages support this construct*, so fewer languages are enabled for this challenge. Check out the Tutorial tab for learning materials and an instructional video!

**Task**

Write a single generic function named *printArray*; this function must take an array of generic elements as a parameter (the exception to this is C++, which takes a *vector*). The locked *Solution* class in your editor tests your function.

**Note:** You must use generics to solve this challenge. *Do not* write overloaded functions.

**Input Format**

The locked *Solution* class in your editor will pass different types of arrays to your *printArray* function.

**Constraints**

- You must have exactly **1** function named *printArray*.

**Output Format**

Your *printArray* function should print each element of its generic array parameter on a new line.

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;
template <class T>
void printArray(vector<T> vec){

   for(int i=0; i<vec.size(); i++)
   cout<<vec[i]<<endl;
}
int main() {
    int n;

    cin >> n;
    vector<int> int_vector(n);
    for (int i = 0; i < n; i++) {
        int value;
```

```
        cin >> value;
        int_vector[i] = value;
    }

    cin >> n;
    vector<string> string_vector(n);
    for (int i = 0; i < n; i++) {
        string value;
        cin >> value;
        string_vector[i] = value;
    }

    printArray<int>(int_vector);
    printArray<string>(string_vector);

    return 0;
}
```