Day 7: Arrays | HackerRank

Terms you'll find helpful in completing today's challenge are outlined below, along with sample Java code (where appropriate).

Data Structures

A way of organizing data that enables efficient storage, retrieval, and use.

A type of data structure that stores elements of the same type (generally). It's important to note that you'll often see arrays referred to as A in documentation, but the variable names you use when coding should be descriptive and begin with *lowercase* letters.

You can think of an array, A, of size n as a contiguous block of cells sequentially indexed from 0 to $n-1$ which serve as containers for elements of the array's declared data type. To store an element, $value$, in some index i of array A, use the syntax `A[i]` and treat it as you would any other variable (i.e., `A[i] = value;`). For example, the following code:

```
// the number of elements we want to hold
final int _arraySize = 4;

// our array declaration
String[] stringArray = new String[_arraySize];

for(int i = 0; i < _arraySize; i++) {
   // assign value to index i
   stringArray[i] = "This is stored in index " + i;

   // print value saved in index i
   System.out.println(stringArray[i]);
}
```

saves and then prints the values listed below in their respective indices of $stringArray$:

```
This is stored in index 0
This is stored in index 1
This is stored in index 2
This is stored in index 3
```

Most languages also have a *method*, *attribute*, or *member* that allows you to retrieve the size of an array. In Java, arrays have a $length$ attribute; in other words, you can get the length of some array, `arrayName`, by using the `arrayName.length` syntax.

Note: The *final* keyword used in the code above is a means of protecting the variable's value by locking it to its initialized value. Any attempt to reassign (overwrite) the value of a *final* variable will generate an error.

Note on Arrays in C++

If you want to create an array whose size is unknown at compile time (i.e., being read as input), you need to create a pointer to whatever data type you'll be declaring your array as (e.g., *char*, *int*, *double*, etc.). Then you must use the new operator to set aside the space you need for your array. The example below shows how to create an array of type *DataType* and unknown size *n* that is read from stdin.

```
// array size
int n;
cin >> n;

// create array of unknown size n
DataType* arrayName = new DataType[n];
```

Additional Language Resources

C++ Arrays

Solve Problem