UCF Local Contest — September 1, 2012

Exact Change

filename: `change`

Whenever the UCF Programming Team travels to World Finals, Glenn likes having the exact amount of money necessary for any purchase, so that he doesn't have to count and receive change. Of course, most countries don't have many different denominations of coins, so Glenn creates different "packages" with him, each with some particular amount of money, in cents. Glenn would like to know which amount of money (in cents), is the smallest that he can't pay for exactly, with some combination of his packages.

The Problem:

Given a list of positive integers, determine the smallest integer that can't be represented as the sum of some subset of the integers on the list.

The Input:

The first input line contains a single positive integer, n $(1 < n \leq 100)$, indicating the number of sets of coin packages to evaluate. Each of the n input sets follows. The first line of each input set contains only an integer, c $(1 \leq c < 31)$, representing the number of different packages of coin for that input set. The following line contains exactly c positive integers, each separated by a single space, representing the value of each of the c packages in cents. The sum of these c integers is guaranteed not to exceed 10^9. Note that the package values are not necessarily distinct, i.e., there may be multiple packages with the same value.

The Output:

For each set of packages, first output "`Set #i: `" where `i` is the input data set number, starting with 1. Follow this with a single positive integer, the smallest value that can't be represented as a sum of the values of a subset of the packages given. Note that a package value can be used at most once in a subset unless there are multiple packages with that value (if there are m occurrences of a package value, up to m occurrences of that value can be used in a subset). Leave a blank line after the output for each test case. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input:

```
3
6
12 8 1 2 4 100
3
1 2 3
6
3 1 3 2 3 3
```

Sample Output:

```
Set #1: 28

Set #2: 7

Set #3: 16
```