Practice Arena

Practice problems aimed to improve your coding skills.

Crazy for Candy

LAB-PRAC-12_FUN-STRUC

Crazy for Candy [20 marks]

Problem Statement
Mr C just loves candy. Actually he is a bit addicted to candy. The situation got so bad that the doctors had to put Mr C on a rehabilitation program and ordered that he must have exactly n grams of candy every day (not one gram less, not one gram more). This was done to restrict his candy intake so that he can get rid of his addiction.

However, candy is only available in the shops in certain weights. We need to help Mr C find all possible ways in which he can take store-bought candy and eat them to fulfill his daily intake. The first line of the input will give you n, the number of possible weights of candy available in the stores. The next line will give you these n weights in increasing order. All the weights will be distinct i.e. no weight will ever repeat. The weights will all be in grams and will all be strictly positive integers, separated by a space. The last line of the input will give you k, the total weight of candy Mr C must consume, in grams.

In your output you have to print all possible ways in which Mr C can consume various numbers of candy of weights given in the input, so that his total candy intake is exactly k grams. Assume that there is unlimited amount of candy of each given weight available in the stores so Mr C can consume as many number of candy of a certain weight as he wants.

If there is no way store bought candy can be consumed in different amounts to get exactly k grams of candy, then print "MR C IS DOOMED" (without quotes) in the output. Otherwise, if one or more combinations are possible, then on every line of the output, you have to print a possible combination of candy that total k grams by printing how many number of candy of each weight Mr C consumes in that combination.

For example, if the stores have two types of candy (i.e. n = 2) of weight 2 grams and 3 grams, and Mr C's daily intake is 6 grams, then he can fulfill his intake in two ways

1. Take 3 candy of 2 grams each. This is represented as "30" (without quotes - no space between the two numbers, no space at the end) indicating that 3 candy of first kind and 0 of second kind are to be taken.
2. Take 2 candy of 3 grams each. This is represented as "02" (without quotes - no space between the two numbers, no space at the end) indicating that 0 candy of first kind and 2 candy of second kind are to be taken.

Your output must print both these combinations on different lines. However, if there are multiple possible combinations, as in the case above, then you must output combinations in a lexicographically decreasing order as described below. We assure you that in none of the combinations Mr C will take, will he every require more than 9 number of any type of candy. Thus, every combination can be written as an n digit number (one digit for every type of candy). You have to output combinations so that these numbers appear in decreasing order. Thus, in the above example, the output should be
30
02
since 30 > 2

Caution

1. Candy weights will all be strictly positive and no two of the n weights we give you will be the same. The n candy weights will be given in increasing order.
2. Be careful to output the combinations in exactly the same order as described in the question. If you output the correct combinations but in the wrong order, then the autograder may give you zero marks since it counts partial marks only when the correct combination is given at the correct location.
3. You have to print the leading zeros in your output as well to indicate that zero number of candy is to be taken. Do not omit the leading zeros. Your output must contain exactly n digits. See the above example, the second combination is printed as 02 not just 2.
4. We will not penalize you for extra new lines at the end of your output but do not have extra spaces at the end of any line of your output.

HINTS: This problem can be solved using recursion. You can write a function of the following kind
void printCombinations(int *weights, int *num, int done, int n, int rem, int *found)

1. The first argument is simply the array of weights of the various types of candy
2. The second argument is a partially filled array to store how many number of candy of each type Mr C should eat
3. The third argument tells us how many types of candy have we already accounted for and how many are left (basically till what point is the array num filled up and from which point onward we should start filling num hereon)
4. The fourth argument tells us how much weight of candy remains to be accounted for. Initially this number is k since we have to account for k grams of candy but as we fill up the num array, this weight will go down.
5. The fifth argument is simply a flag to signal whether we found a single combination or not. Set *found = 1 the moment you find even one valid combination. If found remains zero throughout, you have to print "MR C IS DOOMED" (without quotes) in the output.

You can start off the recursion process, for example, as follows
int found = 0;
int num[n];
printCombinations(weights, num, 0, n, k, &found);

EXAMPLE 1:
INPUT
2
2 3
6

OUTPUT:
30
02

Explanation: See above example.

EXAMPLE 2:
INPUT
```
3
2 4 6
9
```

OUTPUT:
```
MR C IS DOOMED
```

Explanation: There is no way to express 9, an odd number, as a sum of a bunch of even numbers. All weights are even numbers so no matter how many of each type of candy Mr C takes, the resultant sum will always be even.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6238)