Practice Arena

Practice problems aimed to improve your coding skills.

Constrained Candy Crush

LAB-PRAC-13_STRUC-NUM

Constrained Candy Crush [20 marks]

Problem Statement
Recall from a lab problem earlier this week that Mr C is under treatment for candy addiction. He was instructed by his doctor to take only k grams of candy every day. Everyday, Mr C goes to the candy store to get the required amount of candy but yesterday, due to end-sem lab exam preparations, he forgot to buy candy. Now he finds that he has only n pieces of candy in his home, all scattered in mixed up.

Help Mr C find out how to get his daily quota from the candy he has at home. The first line of the input will give you two strictly positive integers n and k, separated by a space. The next line will give you n strictly positive integers, separated by a space. These integers will not be in any specific order but will not repeat i.e. no integer will be given to you twice. These integers represent the weights of the candies Mr C has at his home, in grams.

In your output you have to print all possible ways Mr C can eat k grams of candy using the candy he has at home. Note that Mr C has only one piece of each candy. Suppose k = 5 and Mr C has 4 pieces of candy of weight 4, 3, 2, and 1 grams. Then clearly there are two ways in which he can eat 5 grams of candy - eat the 4 gram and the 1 gram candy (represent this as the string 1001 to indicate that the first and the last candy are to be eaten) and eat the 3 gram and the 2 gram candy (represent this as the string 0110 to indicate that the second and the third candy are to be eaten).

Thus, every combination looks like a long number with 0 and 1 digits. Print these numbers in decreasing order i.e. in the above example, your output should be
1001
0110

If there is no way to eat k grams of candy using the candy provided, print "MR C IS DOOMED" in the output.

Caution

1. Mr C cannot break any candy into two - then creating combinations, the entire candy must be eaten or the entire candy must be set aside.
2. Mr C needs exactly k grams of candy - not one gram more, not one gram less.
3. Note that candy are not given to you increasing order of weights - they are all mixed up.
4. We will not penalize you for extra newlines at the end of your output but do not have extra spaces at the end of any line of your output.

HINTS: Solve this problem using recursion. Write a function of the form
void getCombinations(int *weights, int *choices, int n, int k, int sum, int pos, int *ptr)

1. weights: array of weights given to you as input
2. choices: did you choose the i-th candy or not?
3. n: number of candy
4. k: total weight of candy to be eaten by Mr C
5. sum: how much weight of candy has already been eaten due to choices already made

6. pos: which position in the weight array (i.e. which piece of candy) is it with respect to which a decision now needs to be made?
7. ptr: a pointer to a integer flag variable - set this flag to 1 (using *ptr = 1) the moment any combination is found which adds up to k grams of candy.

Initially, the choices array is all zeros. Gradually fill it up with 1 by choosing various pieces of candy. You may invoke this function as follows:
int isFound;
getCombinations(weights, choices, n, k, 0, 0, &isFound);

EXAMPLE:
INPUT
4 5
4 3 2 1

OUTPUT:
1001
0110

Explanation: see example above.

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/6263)