FRISCO First Bytes High School Programming Contest
10-14-2017
Advanced Category

- **Time Allowed: two hours.**
- **Each team must only use one of the high school's computers.**
- **Answer the questions in any order.**
- **Use only Java 1.8, minGW g++, or MS Visual Studio 14 C/C++**
- **Your program source code must be contained in one source file.**
- **Do not use the "package" construct in your Java code.**
- **Your programs must read from a named file and output to System.out (cout for C/C++ programmers.)**
- **Do not access the web.**
- **Do not use any recording device containing code.**
- **Your solutions must be entirely yours, typed by you during the time allowed for the contest.**
- **As soon as you have solved a problem, submit ONLY the source file via your `PC^2` client.**

Scoring

Equal points will be awarded to each question, even though they may not be equally difficult.

In order to break ties, we will use penalty points. The penalty points for a question will be zero if the question is not answered correctly. If a correct submission occurs for a question at time T minutes, then T penalty points will be added, plus 20 penalty points for each incorrect submission for that question.

A Real Estate

Data File: A.txt
Time allowed = 10 seconds

After the devastating tornado of 2099, one side of Unlucky Avenue in the windy city of No Hope was left in ruins. All of the affected home owners sold their lots to a real-estate developer and moved to the city of Tranquility. The developer has put up the lots for sale. She is willing to sell any number of neighboring lots to the first buyer with the right amount of cash. If buyer Joe with X dollars comes along, what is the maximum number of lots that he can buy?

You are given a list of the prices of the houses for sale, starting with the price of lot 1, then the price of lot 2, and so on.

You must choose the largest contiguous sub-sequence of those lots that Joe can afford.

The total price of the lots that you choose must be less than or equal to Joe's bid of X dollars.

For example, if lot prices are \$10,000, \$20,000, \$30,000, \$20,000, \$30,000, in address order, and Joe is willing to spend \$50,000, Joe can purchase at most two lots.

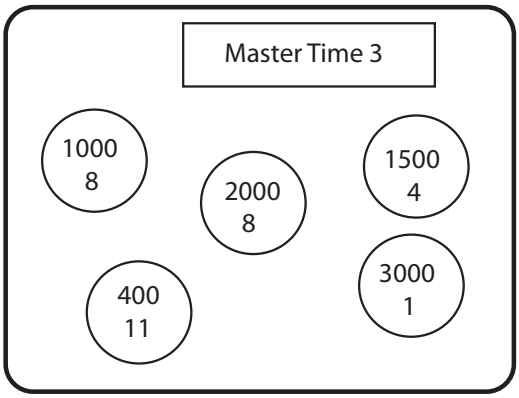
Input:
The input file begins with integer T, $(1 < T < 100)$, on a line by itself, giving the number of test cases.
Each test case begins with a line containing two integers, X and N, separated by a single space. X is Joe's bid and N, $(1 \leq N \leq 100)$, is the number of lots for sale. A line follows containing N space-separated integers, giving the lot prices in order of the lot addresses. The values of X and the lot prices will be integers between 100 and 100000 inclusive.

Output:
For each test case print one line containing the maximum number of lots that Joe can buy.

Sample Input	Sample Output
2 50000 6 10000 20000 30000 20000 30000 21000 7 10000 12000 8000 6000 6000 5000 4000	2 4

B Whack a Money Bag (before it disappears)

Data File = B.txt
Time limit = 10 seconds



snapshot of a game with 3-seconds left to play

A new game has just been installed in the Felix and Flusters Video Arcade. It looks a bit like the famous Whack-a-Mole game, but it's entirely electronic (no moving moles). When you insert the correct amount of money (it isn't cheap) multiple circular icons appear on the screen, each one showing a points value, and an integer counter. At the top of the screen is a Master Timer, M, that shows how many seconds are left before the game ends. It is initially set to some arbitrary integer value, $(1 \leq M \leq 100)$.

Once the game begins the master timer and all of the counters in the icons start counting down synchronously, by one unit per second. When a counter reaches zero, its icon immediately disappears. No new icons are added after the game begins.

During the game you press the icons to get their points values. But you can only press one icon per second. Once pressed, the icon vanishes. You have to sequence your selections in order to build the maximum total points value.

Write a program that computes a maximum points value for a given game.

Input:
The input file will begin with a line containing an integer T giving the number of games to follow, $1 \leq T \leq 20$. The data for each game begins with a line containing two space-separated integers, N giving the number of icons, and M the master timer value.

Then N lines follow, $(1 \leq N \leq 100)$, each containing two space-separated integers, P_i, the points value, and C_i the initial value of the counter for that icon, $(10 \leq P_i \leq 10000)$.

Output:
For each dataset output a single line containing the maximum possible points value obtainable for that game.

Sample Input	Sample Output
2 4 4 1000 1 2000 2 500 2 1200 1 3 4 1000 3 2000 1 500 1	4200 3500

C Stamping Art

Input File: C.txt
Run Time Limit: 10 sec

You are given a large sheet on paper that is divided into r rows by c columns of equal sized squares, and three color-stamps, for Red, Green, and Blue.

Each color stamp prints a square pattern corresponding to two-by-two of the squares on the paper. You can stamp any of the color-stamps on any 2x2 grid of squares on the paper. Some squares on the paper may be left white (not colored) - there is no white color-stamp. Given a pattern, can you reproduce it by using color-stamps?

Input:
The input will begin with a line containing T, $(1 \leq T \leq 20)$, the number of test cases. Each test case begins with a line containing two integers, R and C, giving the numbers of rows and columns on the paper, $(1 \leq R, C \leq 20)$. R lines follow, each containing C characters chosen from the set $\{R, G, B, W\}$, meaning Red, Green, Blue, and White respectively.

Output:
For each test case, print the the word "yes" or "no" according to whether the pattern can be produced.

Sample Input	Expected Output
3	yes
6 5	yes
BBWRR	no
BRRGR	
BRRGW	
BBRRR	
BRGGR	
BBGGW	
3 3	
GGG	
GGB	
BBB	
3 3	
GGG	
GBB	
BGB	

D Almost circular

Data File: D.txt
Time allowed = 10 seconds

We can find an approximation to π by drawing a regular polygon of n sides outside of a circle of unit diameter such that each side of the polygon touches the circumference of the circle, but does not cross it. The perimeter of the polygon is then an approximation to π. Given n, the number of sides of the polygon, compute the error in the approximation to π , accurate to 4 decimal places, by using the equation,

$$\epsilon = \left| \frac{\pi - perimeter}{\pi} \right|$$

Input:
The input will contain between one and twenty datasets, inclusive. Each dataset will comprise a line containing an integer n, the number of sides in the polygon, ($3 \leq n \leq 1000$). The input data will be terminated by a line containing the integer zero. Do nor process this line.

Output:
For each dataset, output the corresponding error rounded to six decimal places.

Sample Input	Sample Output
4	0.099684
3	0.173007
0	

E It's Race Time

Data File: E.txt
Time allowed = 10 seconds

You are to write a program to simulate the events in a car race. Your program will maintain the order of the cars and output an ordered list of the remaining cars at the end of the race.

Input:
There will be at least one and as many as 20 datasets. Each dataset will begin with a `START` command and end with an `END` command. In between these two commands multiple other commands will occur, chosen from the set {`DROPOUT, OVERTAKE, PITSTOP, PITRETURN, CRASH` }.

Data File Commands
Below are the commands that your program must interpret, Values of x are integer car numbers, $10 \leq$ x (or xi) ≤ 999.

```
START n x1 x2 x3  // where there are n cars that begin in order x1, x2, x3, . . .,
                  // where x1 is initially in the lead

DROPOUT x         // car x leaves the race and will not return

OVERTAKE x        // car x overtakes the car in front of it
                  // (there will always be a car ahead of x)

PITSTOP x         // car x temporarily leaves the race

PITRETURN x y     // car x returns to the race in pos'n y (0 = leader)

CRASH x y z . . . // cars x, y, z, . . . permanently leave the race

END               // end of the race - report the remaining cars in place order,
                  // beginning with the winner, as a space-separated list
                  // of integers on one line
```

`n` is the number of cars initially in the race, $1 \leq$ `n` ≤ 1000

Throughout the race only one car will occupy each position or place.

Cars that are in the pits when the E command is read will not be present in the final list output by the program.

The input file is terminated by End of File.

Sample Input Data	**Output for Sample Input**
START 10 3 16 2 7 4 9 1 100 55 87 DROPOUT 100 OVERTAKE 16 OVERTAKE 3 OVERTAKE 7 CRASH 16 1 7 PITSTOP 3 PITSTOP 9 PITRETURN 9 4 PITSTOP 2 PITRETURN 3 3 END	4 55 87 3 9

F Graduation

Data File: F.txt
Time allowed = 10 seconds

Write a program for the principle of Average High School that she can use to make a list of the graduates ordered according to the students' names, GPAs and ID numbers.

Input:
There will be only one dataset. It begins with an integer N on a line by itself, giving the number of records to follow, ($1 \leq N \leq 1000$). N lines follow formatted as described below:

Name (between 10 and 40 characters),
GPA (one of A+, A, A-, B+, B, B-, C+, C, C-, D+, D, D-),
ID Number (a 10 digit integer)

The three fields will be separated by single spaces, but note that there could also be a hyphen within a student's family name, and some students will have several given names. A comma will follow the family name of each student and the given names will be space-separated (as shown below)

Output:
Output a list of the N student records ordered first alphabetically by the students' names. If two or more students have exactly the same name their records must be ordered by their GPAs, in the order of highest GPA to lowest. And if two or more students have the same names and GPAs their records must be listed in increasing order of their ID numbers. Each record output will have the name first, padded to 40 characters by adding spaces to the right of the names, a single space, then the student's GPA, padded to two characters by adding a space character where necessary, then another space, then the student's ID number, which is always ten digits long.

Sample Input

```
7
Chicken, Alfred James B+ 1032043056
Archer, Bill A- 1357258896
Juniper, Jane Janice June B+ 2010305071
Cedar-Oak, Brian A 1239874237
Milagro, Juanita Maria Josea Alejandra A+ 7456312985
Juniper, Jane Janice June A- 2010305071
Archer, Bill A- 1357258885
```

Output for Sample Input

```
Archer, Bill A- 1357258885
Archer, Bill A- 1357258896
Cedar-Oak, Brian A 1239874237
Chicken, Alfred James B+ 1032043056
Juniper, Jane Janice June A- 2010305071
Juniper, Jane Janice June B+ 2010305073
Milagro, Juanita Maria Josea Alejandra A+ 7456312985
```

H The Divider

Data File: H.txt
Time allowed = 10 seconds

Each week John takes a sack of wheat to the miller, who grinds it into fine flour. Today he brought 100 lbs of wheat. Other weeks it might be more or less, but always an integer amount. Then he goes to The Divider, a machine that separates the flour into two amounts as instructed on the control panel. John pours his 100 lbs of flour into the machine and presses 80. He gets two bags, one with 80 lbs, the other with 20 lbs. He puts the 80 lbs back into the machine and presses 50, getting two bags weighing 50 lbs and 30 lbs. Finally he puts the 30 lbs back into the machine and presses 20, getting 20 lbs and 10 lbs. He takes the 50 lbs, 30 lbs, 20 lbs, and 20 lbs sacks to deliver to his relatives, whose families' needs match these amounts.

The bill for the divider is 100 + 80 + 30 = 210 glicks, calculated by adding together the weights of the amounts John put into the machine. He is mystified by the way that the bill is calculated and is sure he could have saved money had he divided his flour differently. For example, had he divided the 100 into 50 and 50, then one of those into into 20 and 30, then the 30 into into 20 and 10, the bill would have been 100 + 50 + 30 = 180 glicks. Can you help?

Input:
The input will begin with integer T on a line by itself giving the number of test cases, $(1 \leq T \leq 20)$. T test cases follow, each comprising a series of space-separated integers on a single line. Each test case begins with an integer W, giving the weight of the sack of flour that John brings to The Divider, $(1 \leq W \leq 1000)$. Then up to 100 integers follow in non-decreasing order, each greater than zero, that sum to W. These are the amounts that John wants to deliver to his relatives.

Output:
For each test case print a single line as shown in the Sample Output below, giving the minimum cost of dividing the flour.

Sample Input	Output for Sample Input
2	The minimum cost is 180 glicks.
100 10 20 20 50	The minimum cost is 190 glicks.
120 30 40 50	