**Problem A: Prime Factorization**

Many encryption algorithms rely on the fact that it is very difficult to factor large numbers. These algorithms use numerical keys that may be hundreds of digits long. It would take several decades to find the prime factors of just one of these numbers. Mathematicians have been trying to create fast factoring algorithms for several years now, without any success.

Fortunately, your assignment is much easier. You will write a program to factor relatively small numbers, up to 1000. Your program should print each number along with the prime factors that make up that number. A prime number is a number that is only divisible by itself and 1. The first 5 prime numbers are 2, 3, 5, 7, and 11. Note that a prime may appear more than once in a number’s list of factors. For example, the prime factors of 8 are 2, 2, and 2.

**Input Specification**

The input will contain a list of integers, each on a separate line. The integers will be between 2 and 1000, inclusive. The list will be terminated by a ‘-1’. (Don’t process the ‘-1’).

**Output Specification**

Your program should print the prime factors for each number specified in the input. This should be formatted as a multiplication in the form “\( M = F_1 \times F_2 \times \ldots \times F_i \)”, where \( M \) is the number to factor, and \( F_1 \times F_2 \times \ldots \times F_i \) are the factors. The factors should be printed in increasing order.

**Sample Input**

```
18
100
13
360
20
-1
```

**Sample Output**

```
18 = 2 \times 3 \times 3
100 = 2 \times 2 \times 5 \times 5
13 = 13
360 = 2 \times 2 \times 3 \times 3 \times 5
20 = 2 \times 2 \times 5
```
Problem B: The Tortoise and the Hare

The tortoise and the hare are having their annual race today. For the first time ever, this race will be broadcast live over the internet. The tortoise and the hare are both very excited about this, and they want to make sure everything goes according to plan. During the race, several statistics will be broadcast. One of these statistics is the current distance between the runners. For this, they need your help.

The tortoise and the hare race several laps around a perfectly circular track with a radius of \( r \) meters. At the start of the race, both contestants start running counter-clockwise around the track. The starting point of the race is at the northernmost point of the track. Both the tortoise and the hare will run at constant (but different) speeds throughout the race.

The distance between the tortoise and the hare will be measured as the direct Euclidean distance. This is the distance of the shortest line segment going from the hare’s position to the tortoise’s position. For example, suppose the tortoise runs at a speed of 6 meters/second while the hare runs at a speed of 9 m/s, and the radius of the track is 10 meters. When the tortoise is half way around the track, the hare will be three quarters of the way around the track. This forms a right triangle, giving a distance \( d \) of \( 10 \sqrt{2} = 14.14 \) meters.

You will be given the radius of the track, the speed of the tortoise, and the speed of the hare. From this you will calculate the distance between the racers at several different times after the start of the race.
Problem B: The Tortoise and the Hare (continued)

Input Specification

The first line of input will contain the radius of the track. This will be a floating-point value between 1.0 and 100.0 meters, inclusive. The next line will contain speed of the tortoise followed by the speed of the hare. These are both floating-point values between 0.0 and 100.0 meters per second, inclusive. The third line will contain an integer N, the number of time values to check. This will be followed by N time values, each on a separate line. The time values are in seconds from the start of the race and will range from 0.0 to 1000.0 seconds, inclusive.

Output Specification

For each time value listed in the input, your program will print the current distance between the tortoise and the hare. This will be in meters and should be rounded to two decimal places. Each value should appear on a separate line.

Sample Input

10.0
6.0 9.0
4
5.236
10.0
2.5
0.0

Sample Output

14.14
19.95
7.33
0.00
Problem C: Word Wrap

You have just been hired by a major software company, and your first assignment is to write the word wrap feature of their new word processor. The word wrap feature breaks up long lines of text into several smaller lines so that the text does not run off the edge of the page. The tricky part is that words in a line of text must be kept intact while doing this. It would look very unprofessional if the program inserted line breaks into the middle of words.

Your program will read in several lines of text. Your program will reformat each line into smaller lines that are no more than 20 characters each. If a word would make the line extend past 20 characters, the entire word should be shifted to the next line. A word consists of a string of characters that does not contain a space. Words may be made up of letters, digits, and/or punctuation marks. You will be guaranteed that no word in the input is larger than 20 characters.

To make the output more concise, extraneous spaces in the input should be ignored. Your program should only output one space in between each word, even if multiple spaces appear in the input.

Input Specification

The input will consist of several lines of text. Each line will contain at most 100 characters. There will be at least one word in each line. The input will be terminated with a line consisting of the text “END”.

Output Specification

Your program should output a formatted set of lines for each line of text in input. A blank line will separate each set of lines.

Sample Input

The quick brown fox jumped over the lazy dog. 1234567890
Did you see the game in Lane Stadium? I hear we won.
LSU fans sob. Virginia Tech fans rejoice. One more victory.
END
Problem C: Word Wrap (continued)

Sample Output

The quick brown fox
jumped over the lazy
dog.

12345678901234567890

Did you see the game
in Lane Stadium? I
hear we won.

LSU fans sob.
Virginia Tech fans
rejoice. One more
victory.
Problem D: How do I get to CS 1044?

The departments of computer science and computer engineering will soon be merging, and they will be building a new academic building. Current plans are to call this building McBryderson. The designers of McBryderson Hall decided to have a little fun, and they designed the building in the shape of a maze. This is unfortunate for the students, as they now have to figure out how to navigate through the building to get to their classes. You need to help them by writing a program that will determine the shortest path between two rooms in McBryderson Hall.

McBryderson Hall is arranged in an N x N grid, where 2 ≤ N ≤ 10. Each square in the grid can contain a wall, a path, or a room. Students can move in 4 directions within McBryderson: north, south, east and west. Students cannot move through walls. Your program should output a path with the minimum number of steps to get from one room to another. Assume the students will not try to leave the building and enter through another entrance. The students may need to travel through one room to get to another. If there are multiple paths that share the same minimal length, any of them may be used.

Input Specification:

The first line of input will be an integer N, the size of the grid. This will be followed by N lines, each line describing one row of the building. Walls are represented by ‘W’, paths are represented by ‘P’, and classrooms are represented by the digits ‘0’-‘9’. After this will be a list of pairs of room numbers, one pair to a line. At least one valid path will exist between the two rooms. The list will be terminated by the pair “-1 -1”.

Output Specification:

For each pair of rooms listed in the input, you will output a sequence of moves to get from the first room to the second room. Each move consists of one of the following: “North”, “South”, “East”, or “West”. Each sequence of moves will be printed on a separate line. The moves within each sequence will be separated by single spaces.

Sample Input:

```
6
WWWWW
W0PWWW
WWPPWW
W1PPWW
WWPWW
WW2WWW
0 2
1 0
1 2
-1 -1
```

Sample Output:

```
East South South South South
East North North West
East South South
```
Problem E: What’s My GPA?

Virginia Tech will soon be adding a more detailed GPA report to their student transcripts. In addition to the student’s overall GPA, the report will give the student’s GPA within each department that he or she has taken at least one class. It is your job to write this functionality.

To compute a GPA, perform the following steps: For each class the student took, multiply the number of credit hours of the class by the number of quality credits earned. The quality credit values for each grade are listed in Table 1. Then sum the total number of quality credits received for all classes. Divide this result by the total number of credit hours.

\[
\begin{align*}
A &= 4.0 \\
A- &= 3.7 \\
B+ &= 3.3 \\
B &= 3.0 \\
B- &= 2.7 \\
C+ &= 2.3 \\
C &= 2.0 \\
C- &= 1.7 \\
D+ &= 1.3 \\
D &= 1.0 \\
D- &= 0.7 \\
F &= 0.0
\end{align*}
\]

Table 1: Quality Credit Values

Input Specification

The first line of input will contain an integer \(N (1 \leq N \leq 100)\), the number of courses the student took. This will be followed by descriptions of \(N\) classes, one class per line. Each line will contain the department the class was taught under, the course number, the number of credits, and the grade the student received. The department will be an upper-case abbreviation between 2 and 4 characters. For example, CS is used for computer science and ENGL represents English. The course number will be a 4 digit number. The number of credits will range from 1 to 9, inclusive.

Output Specification

The first line of output will contain the overall GPA, in the form “Overall GPA: \(<G>\)”, where \(<G>\) is the GPA rounded to 2 decimal places. This will be followed by the GPAs for each department in which the student took at least one class. These should be displayed one to a line, sorted alphabetically by department. These lines should be formatted as “\(<\text{Dept}>\) GPA: \(<G>\)”, where \(<\text{Dept}>\) is the department abbreviation and \(<G>\) is the GPA rounded to 2 decimal places.
Problem E: What’s My GPA? (continued)

Sample Input

10
ENGL 1105 3 B
CS 1044 3 A
HIST 1115 3 B+
CHEM 1015 3 C-
CHEM 1025 1 A-
CS 1704 3 B+
CS 2704 3 A-
HIST 1116 3 C
MATH 1114 2 B+
MATH 1224 2 D+

Sample Output

Overall GPA: 2.92
CHEM GPA: 2.20
CS GPA: 3.67
ENGL GPA: 3.00
HIST GPA: 2.65
MATH GPA: 2.30
Problem F: How do I pack my backpack?

Freshman students at Virginia Tech are having problems packing their backpacks for their classes. They have not yet learned that things such as textbooks are poor choices to pack, as they are very heavy and not very useful in class. They also have cheap backpacks that can only hold up to a certain weight. A senior in Computer Science was nice enough to write a program that would generate the best combination of items to carry for a given a weight limit. The only problem was that the program was written in Cobol and does not run on any modern computer platform. It is your job to write a new version of this program.

Your program will read in a list of items, along with their weights and associated value ratings. A value rating is how useful an item is in class, on a scale of 1 to 100. For example, a pen will generally be more useful than a pair of computer speakers in class. Thus the pen may have a rating of 90 while the speakers are rated at 10. Your program will generate a list of items to carry that maximizes the total value without exceeding the maximum weight limit. All weights are measured in ounces.

If two or more solutions are tied for the highest value rating, you should choose the one with the lowest weight. You will be guaranteed that this will produce a unique solution.

Input Specification

The first line of input will contain an integer $N$ ($1 \leq N \leq 10$), the number of data sets to consider. Following this will be a description of each data set. Each data set will list the maximum weight $M$ ($1 \leq M \leq 1000$), in ounces, followed by the number of items in that data set. These will appear on the same line, separated by a space. There will be between 1 and 20 items, inclusive. The list of items will appear after this, with one item to a line. Each line in the item list will contain the weight of the item, the value of the item, and the name of the item, separated by spaces. The weight and value are both integers from 1 to 100, inclusive. The name will contain at most 20 characters, consisting of letters and underscores (’_’s). Remember to treat each data set separately.

Output Specification

Your program should produce output for each data set contained in the input. The first line printed will identify which data set is being printed, in the form “Data Set I”, where I is the number of the data set, starting with 1. This should be followed by the list of items to pack, one item per line. This should be formatted in the same way as the input, with the weight, followed by the value, followed by the name. The order of items is not important. After this, a summary line should be printed for each data set, in the form “Total Weight: \textbf{W}, Total Value: \textbf{V}”, where \textbf{W} is the total weight of the items printed and \textbf{V} is the total value. Each data set should be separated by a blank line.
Problem F: How do I pack my backpack? (continued)

Sample Input

3
200 4
100 10 calculus_book
2 95 pen
10 97 notebook
100 40 laptop_computer
100 3
40 100 group_project_report
40 50 ergonomic_keyboard
30 50 graphing_calculator
200 2
4 90 blue_book
5 100 lucky_pencil

Sample Output

Data Set 1
2 95 pen
10 97 notebook
100 40 laptop_computer
Total Weight: 112, Total Value: 232

Data Set 2
40 100 group_project_report
30 50 graphing_calculator
Total Weight: 70, Total Value: 150

Data Set 3
4 90 blue_book
5 100 lucky_pencil
Total Weight: 9, Total Value: 190