November 2010

Computer Science Competition
Hands-On Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Point Values and Names of Problems

Number	Name
Problem 1	Clothes
Problem 2	Friends
Problem 3	Periods
Problem 4	Population
Problem 5	Presidents
Problem 6	ReWord-Wrap
Problem 7	Seasons
Problem 8	Semiperfect
Problem 9	Spaces
Problem 10	Stacker
Problem 11	Star
Problem 12	Trail

1. Clothes

Program Name: Clothes.java **Input File: clothes.dat**

Billy's mom recently washed his clothes. When his mom finishes washing his clothes, she folds them and stacks them in one large pile. It is then Billy's responsibility to sort said clothes into three drawers, these being shirts, pants, and socks. When he places these items in the drawers he takes the top one off of the stack his mom gave to him and places it at the top of the stack in the respective drawer.

Each day Billy must wear a shirt, pants, and socks. He simply picks the top one off of each stack in each drawer to wear for that day. Given the original stack of washed clothes, determine what Billy will wear up until he needs more clothes washed. Billy needs more clothes when he runs out of any necessary article of clothing for the day. Billy will only wear a piece of clothing once before it is declared dirty and needs washing. He does not ever wear dirty clothes.

Input

The first line of input will contain a single integer `n` that indicates the number datasets to follow. Each dataset begins with an integer `c` which is between 1 and 50. The next `c` lines are the stack of clean clothing in the format "*`article (type)`*" where *`article`* is the name of the article of clothing and *`type`* is the type of clothing. The value of *`type`* will be `shirt`, `pants`, or `socks`.

Output

The output will be a series of lines. Each line will contain the three articles of clothing for the specific day separated by commas. The shirt will be printed first followed by the pants and socks. The type should not be printed. There should be a blank line between the output of each dataset.

Example Input File

```
2
6
mario logo (shirt)
red  (pants)
hawaiian (shirt)
baggy (pants)
white (socks)
fancy (socks)
10
khaki (pants)
blueish green (shirt)
apple (shirt)
bright (socks)
blue (pants)
red (pants)
purple (socks)
ugly (shirt)
stupid (socks)
big red (pants)
```

Example Output to Screen

```
hawaiian, baggy, fancy
mario logo, red , white

ugly, big red, stupid
apple, red, purple
blueish green, blue, bright
```

2. Friends

Program Name: Friends.java **Input File: friends.dat**

Jane has many friends, but she likes some less than others. She has a number associated with each friend which represents how much she likes each friend. The higher the number, the more she likes that friend. Given her list of friends, sort them so that the best friends are at the top and the least best friends are at the bottom.

Input

The first line of input will contain a single integer `n` that indicates the number datasets to follow. Each dataset will consist an integer `m`, (0 < `m` < 500), which is the number of friends that will be in the dataset. The next `m` lines will of one line in the format "*name num*", where *name* is the name of the friend and *num* (0 < *num* < 500) is the number that represents how much she likes her friend. The name will only be one word. No two friends will not have the same *num* value.

Output

Output a comma separated list of names of the friends in order of how much Jane likes them.

Example Input File

```
2
2
bill 2
greg 5
4
jim 5
phil 8
paul 7
rob 2
```

Example Output to Screen

```
greg, bill
phil, paul, jim, rob
```

3. Periods

Program Name: Periods.java **Input File: periods.dat**

Billy gets distracted so sometimes he forgets to put periods at the end of his sentences. To help him out, you are to put a period at the end of his sentences if the period is not already present.

Input

The first line of input will contain a single integer `n` that indicates the number of lines to follow. Each line will consist of a sentence which may or may not have a period at the end.

Output

Output the sentence, making sure there is one period at the end.

Example Input File

```
3
You kicked my dog
No I did not.
It was the kid that did
```

Example Output to Screen

```
You kicked my dog.
No I did not.
It was the kid that did.
```

4. Population

Program Name: Population.java **Input File: population.dat**

In Robert's country it is estimated that a person dies every 7 seconds, and a person is born every 4 seconds. Given a beginning population, estimate the population after a certain period of time.

Input

The first line of input will contain a single integer `n` that indicates the number of lines to follow. Each line will consist of two integers, `p` and `t`, where `p` is the beginning population, and `t` is the amount of time that will pass. Both `p` and `t` will be between the number 1 and the number 2 billion.

Output

Output the estimated population after the period of time based on the above statistics.

Example Input File

```
3
12 14
530 200
4786 3543
```

Example Output to Screen

```
13
552
5165
```

5 Presidents

Program Name: Presidents.java **Input File: presidents.dat**

A log is kept of the bills in the wallet, but in the log it does not indicate the actual value of the bills. It simply states the last name of the president on the front of each bill. You are to determine the actual value of what is in the wallet.

Input

The first line of input will contain a single integer `n` that indicates the number of lines to follow. Each line will consist of a series of last names of presidents. The presidents that could appear are: `Franklin`, `Grant`, `Jackson`, `Hamilton`, `Lincoln`, and `Washington`. The values of these presidents are $100, $50, $20, $10, $5, and $1, respectively.

Output

Output the value of the money in the wallet with a dollar sign in front.

Example Input File

```
3
Franklin Grant Jackson
Hamilton Lincoln Washington
Washington Washington Washington Franklin Jackson
```

Example Output to Screen

```
$170
$16
$123
```

6. ReWord-wrap

Program Name: ReWordWrap.java **Input File: rewordwrap.dat**

When a window of a text editor resizes, it is often necessary to readjust all of the containing words to fit in the screen window without splitting up any of the words. This is referred to as word-wrapping. Given a block of text, word-wrap the data to fit in a specific size.

Input

- The first line of input will contain a single integer `n` that indicates the number of datasets to follow. Each dataset will consist of:
    - An integer `w` indicating the number of characters to word wrap to where $10 \le w \le 50$.
    - An integer `m` indicating the number of lines of text input that are to be word wrapped where $1 \le m \le 100$.
    - The `m` lines of text is what is to be word wrapped.

Output

Output the text word wrapped to the proper number of characters.

Example Input File

```
2
40
4
Chinua Achebe was born on November 16, 1930 into the Igbo
African ethnic group in southern Nigeria.  He was raised in a
Christian family, but he was always fascinated by the
traditional African religion and culture.
35
4
Achebe uses the short story of a hard working man
in a war torn area with a bright outlook on life to act
as a model for other Africans affected by
the Nigerian Civil War.
```

Example Output to Screen

```
Chinua Achebe was born on November 16,
1930 into the Igbo African ethnic group
in southern Nigeria.  He was raised in a
Christian family, but he was always
fascinated by the traditional African
religion and culture.

Achebe uses the short story of a
hard working man in a war torn area
with a bright outlook on life to
act as a model for other Africans
affected by the Nigerian Civil War.
```

7. Seasons

Program Name: Seasons.java **Input File: seasons.dat**

Jenkins is travelling on a long journey to a far away land by foot. He moves at different speeds at different times of the year due to different weather patterns. More specifically, he can move 3 miles a day during the hot season of January through April, 5 miles a day during the warm season of May through August, and 1 mile per day during the cold season of September through December. Given a starting date, and the distance Jenkins must travel, calculate when Jenkins will reach his destination.

Input

- The first line of input will contain a single integer `n` that indicates the number of datasets to follow.
- Each dataset will be composed of:
  - An integer `d` between 1 and 12000, which is the distance Jenkins must travel.
  - A date in the format "*m d, y*", where *m*, *d*, and *y* are the month, day of the month, and year, respectively, on which Jenkins begins his journey. All dates will be valid.

Output

Output the valid date that Jenkins will arrive at his far away destination in the format "*m d, y*", where *m*, *d*, and *y* are the month, day of the month, and year, respectively.

Example Input File

```
3
1000
January 1, 1999
365
August 12, 2034
712
December 31, 1991
```

Example Output to Screen

```
September 26, 1999
February 17, 2035
July 09, 1992
```

8. Semiperfect

Program Name: Semiperfect.java **Input File: semiperfect.dat**

A semiperfect number is a number that all or some of the divisors (not including itself) of the number can be added together to get the original number. For instance, the number 12 has the following applicable divisors: 1, 2, 3, 4, 6. The divisors 6, 4, and 2 can be added together to get 12, therefore 12 is a semiperfect number. Given a number, determine whether or not it is semiperfect.

Input

The first line of input will contain a single integer `n` that indicates the number of lines to follow. Each line will consist of a single integer `m` (1 < `m` < 1000000), which will be the number that you are required to determine to be semiperfect or not semiperfect.

Output

If `m` is semiperfect print "`Semiperfect`", otherwise, print "`NOT Semiperfect`".

Example Input File

```
4
3
6
32
228
```

Example Output to Screen

```
NOT Semiperfect
Semiperfect
NOT Semiperfect
Semiperfect
```

9. Spaces

Program Name: Spaces.java **Input File: spaces.dat**

For some reason, when Paul's essay was saved, all of the spaces disappeared. He does not want to re-add all of his spaces so he decided to make a program that will, given a dictionary of words, add the spaces in the right places. There will only be one configuration of spaces that will correctly separate the words.

Input

- The first line of input will contain a single integer `n` that indicates the number of words in the dictionary.
- The following n lines will be the words that are in the dictionary of possible words.
- The rest of the input will be the collection of characters that must be spaced out correctly. In the data file this will be on one line.

Output

Output the zero-based index positions of the spaces in the final essay.

Example Input File

```
12
there
is
nothing
i
an
and
a
the
now
know
sure
do
thenowisnothingandisuredoknowtheis
```

Example Output to Screen

```
3 7 10 18 22 24 29 32 37 41
```

10. Stacker

Program Name: Stacker.java **Input File: stacker.dat**

The goal of this problem is simple; given a set of two-dimensional blocks, determine the least number of blocks necessary to construct them in a given configuration, if possible. All blocks have a width of 1, but the length will vary. The blocks themselves are stacked similar to Tetris, in that they are dropped from the top and will fall until any part of the falling block has collided with the preexisting blocks, and they can be rotated.

You are limited to the number and lengths of blocks provided by the data. Not all configurations can be achieved with the given blocks.

Input

The first line of input will contain a single integer `n` that indicates the number of data sets to follow. Each data set will consist of:

- A line containing two integers `r` and `c`, indicating the number of rows and columns, respectively, that the configuration will use. The value of `rows` and `cols` will both be between 1 and 50, inclusive.
- The next `r` lines of `c` characters will be the configuration that you are trying to achieve, where a `.` (period) represents an open area and a `#` is a portion of a block.
- The next line will contain a series of integers between 1 and 50 (inclusive), which represent the lengths of the blocks available for stacking.

Output

If it is possible to stack the available blocks in the given configurations, print the least number of blocks that could be used to accomplish this. If it is not possible, print "`Not Possible.`"

Example Input File

```
2
5 5
.....
#####
...#.
...#.
#..#.
1 3 2 1 4
7 7
.......
....###
.#...##
.#...#.
###..#.
..#..#.
..#..#.
1 2 2 2 2 3 3 4 5
```

Example Output to Screen

```
Not Possible.
6
```

11. Star

Program Name: Star.java **Input File: none**

Print the star.

Input
None

Output
Print the star-like shape exactly as shown below.

Example Output to Screen
```
.........*.........
.........**........
.........*.*.......
...******...*......
....*........******
.....*...........*.
.....*..........*..
....*..........*...
...*....*.....*....
..*...*..*...*.....
.*..*......*..*....
.**..........*.*...
...............**..
```

12. Trail

Program Name: Trail.java **Input File: trail.dat**

Jim works for a shipping company in the wild frontier. His job is simple; he must deliver a package to the West before a certain deadline. You are to determine when Jim will reach his destination and whether or not Jim is late.

You will be given a top-down map of the area Jim must traverse and the number of days until the deadline that Jim must deliver the package by. It will take Jim 3 days to cross a normal area of land, 5 days for him to cross a wooded area, and 8 days for him to cross a body of water. Jim cannot cross over rocks because it will break his wagon. His goal is to get to the West, which is simply the far left column on the map. He can travel in any of the four cardinal directions (no diagonals). On some maps it may be impossible for Jim to reach the West.

Input

The first line of input will contain a single integer `n` that indicates the number of data sets to follow. Each data set will consist of:

- A line containing two integers: `rows` and `cols`, indicating the number of rows and columns, respectively. The value of `rows` and `cols` will both be between 1 and 50, inclusive.
- The next `rows` lines of `cols` characters will be the map of the area. Each character will be one of the following:
    - `.` (period) – a normal area of land
    - `W` – an area of water
    - `R` – a rock
    - `F` – a wooded area
    - `S` – Jim's starting location, there will only be one of these
- The next line will contain a single integer, between 1 and 300 inclusive, which represents the number of days until the deadline.

Output

You are to output the result of Jim's trip. If Jim is able to get to the West by the deadline print "`The Package arrived with` *`d`* `day(s) to spare.`" where *`d`* is the number of days still left until the original deadline. If Jim cannot deliver the package in time, simply print "`The Package was not delivered in time.`"

Example Input File

```
2
7 7
..W....
..W....
..W....
..W..FF
..W.R..
..W..R.
.....RS
35
5 10
....RF....
....RF....
...R.FF..S
..R..FF...
...R.FF...
50
```

Example Output to Screen

```
The Package arrived with 4 day(s) to spare.
The Package was not delivered in time.
```