SOEN 6501 **PCT SESSION #1** **Feb 18, 2017 (10:00 – 12:30hrs)**

Student Id: **Name:** **Signature:**

Login to the PC^2 Lab System. Refer to the *SOEN6501LabInstructionForStudents* document for instructions. You can also download the version available on the *H:* drive and save it on your desktop.

Please note that your programs MUST produce the CORRECT results for all test cases within PC^2, including spaces and newlines as specified in the problem. Some of the test cases are generated internally and are not revealed to you.

Reminder:

When submitting your solution to the lab system, make sure there is no `package` statement at the top of your `.java` file as this will result in a grade of 0 (restriction of this system).

Transcript Name Printing (Basic Problem)

Problem Statement:

The student enters her/his name in the order of FIRSTNAME MIDDLENAME SURNAME (also known as family name). But the College has decided to print name on the student transcript as SURNAME FIRSTNAME MIDDLENAME. You have to write a JAVA program to do this required manipulation.

But since student population comes from all over the world, it has many different types of names. Some students don't have all 3 components; they just use FIRSTNAME and MIDDLENAME or SURNAME. Some even do not use MIDDLENAME and SURNAME both. So, in such cases, the challenge is that the program has to change the order of names only if the student enters name as FIRSTNAME SURNAME and keep sequence as it is if the student enters the name as FIRSTNAME MIDDLENAME. To handle cases like this, upon study, the university has decided to consider all names ending with vowels (a,e,i,o,u or A,E,I,O,U), only in such cases, as SURNAMES. If student's name does not have MIDDLENAME and SURNAME both, student's name will be printed as it is.

Input Specification:

The name is input in the format FIRSTNAME MIDDLENAME SURNAME or FIRSTNAME SURNAME or FIRSTNAME MIDDLENAME or FIRSTNAME. FIRSTNAME, MIDDLENAME and SURNAME each are separated by a single space. Each component can contain maximum 50 characters. Names are case sensitive. Name is terminated by a newline character.

Output Specification:

Name as expected SURNAME FIRSTNAME MIDDLENAME or SURNAME FIRSTNAME or FIRSTNAME MIDDLENAME or FIRSTNAME. Names MUST be separated by a single space between them. There should be no space at the beginning of the line and the output should be terminated by newline character without any space before it.

Sample Input Output:

Input A:
Prithviraj Dajisaheb Chavan
Output A:
Chavan Prithviraj Dajisaheb

Input B:
Prithviraj D. Chavan
Output B:
Chavan Prithviraj D.

Input C:
Barack Obama
Output C:
Obama Barack

Input D:
Michael Jackson
Output D:
Michael Jackson

Note: There are 12 test cases named A to L for this problem. The first four, A, B, C and D are revealed to you above. You must submit your program for each test case separately, as Problem 1 (A), Problem 1 (B), …, Problem 1 (L). If your program works correctly for all 12 cases, you have succeeded in solving this programming question.

Reminder:

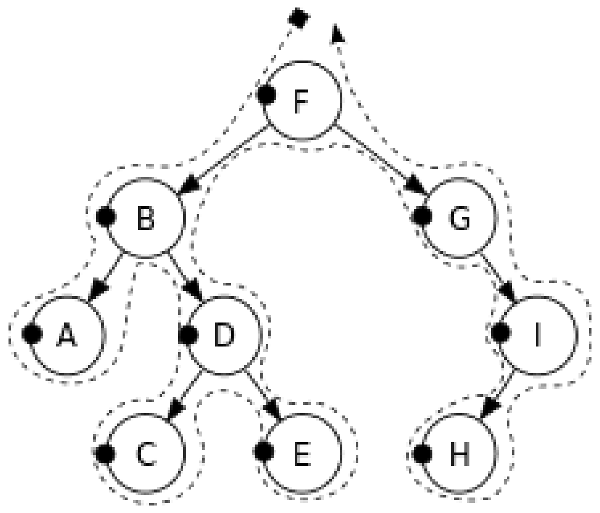
When submitting your solution to the lab system, make sure there is no `package` statement at the top of your `.java` file as this will result in a grade of 0 (restriction of this system).

Family Tree Relationship (Advanced Problem)

Problem Statement:

A family tree is a chart representing family relationships in a conventional hierarchical tree structure with nodes and branches connecting to other nodes. Family trees are often presented with the oldest generations at the top and the newer generations at the bottom.

Family tree can have many relations. Given the family tree and a relation among two family members you need to write a program to print whether the relation is true or false, and nodes traversed in a pre-order.



Pre-order: F, B, A, D, C, E, G, I, H.

Pre-order traversal is defined as follows:

1. Check if the current node is empty / null.
2. If not null display the data part of the root (or current node).
3. Traverse the left subtree by recursively calling the pre-order function.
4. Traverse the right subtree by recursively calling the pre-order function.

Definitions:

1. **Parent:** Node B is parent of A, if B and A are directly connected and A is exactly one level below B in the family tree. Node A is **child** of Node B.
2. **Sibling:** Siblings are members that have the same immediate parent.
3. **Descendant:** Node A is a descendant of Node B, if A is connected to B either directly or through any other node but is at least one level below B in the family tree.
4. **Ancestor:** Node B is an ancestor of Node A, if B is connected to A either directly or through any other node but is at least one level above A in the family tree.

Input Specification:

1. The first line contains an integer n (0 < n < 100) followed by 'n' data sets.
2. Each data set consists of two strings, separated by space. The first string indicates the parent of the second string. Each string in the data set won't exceed more than 25 characters. Strings are NOT case sensitive.
3. An integer m (0 < m < 100) indicating number of relations for the family.
4. The following 'm' lines describe the relation in the family (**child, parent, sibling, ancestor, descendant**).

Assumptions:

1. No name appears more than once in the family tree.
2. Each parent can have at most 2 children.
3. If a person has only one child then the child will be the left child of that node in the family tree.
4. The first string in the data set will be the root element.
5. Except for the root, nodes can only be added to the tree if the parent is already present in the tree.

Output Specification:

1. For each relation in the data set, your program should output T or F indicating whether the relation is true or false respectively. There should be no space at the beginning of any line and there should be a single space between relations. The output should be terminated by a new line character without any space before it.
2. Any relation with names not appearing in the family tree should result in F.
3. Print the preorder traversal of the tree on the next line with a single space separating each name.
4. The output should be terminated by a new line character without any space before it.

Sample Input Output:

Input A:

```
8
Motilal Jawahar
Jawahar Indira
Motilal Kamala
Indira Sanjay
Sanjay Varun
Indira Rajiv
Rajiv Priyanka
Rajiv Rahul
6
Motilal child Jawahar
Varun descendant Indira
Priyanka sibling Varun
Sanjay child Indira
Sanjay ancestor Varun
Kamala ancestor Rahul
```

Output A:

```
F T F T T F
Motilal Jawahar Indira Sanjay Varun Rajiv Priyanka Rahul Kamala
```

Input B:

```
9
Prithviraj Raj
Shashi Sanjana
Prithviraj Shashi
Raj Randhir
Rishi Ranveer
Randhir Bebo
Randhir Lolo
Raj Rishi
Rishi Ridhima
7
Bebo descendant Shashi
Raj sibling Shashi
Prithviraj ancestor Ridhima
Lolo sibling Ridhima
Bebo ancestor Shashi
Prithviraj ancestor Raj
Rishi descendant Raj
```

Output B:

```
F T T F F T T
Prithviraj Raj Randhir Bebo Lolo Rishi Ridhima Shashi
```

Note: There are 8 test cases named A to H for this problem. The first two, A and B are revealed to you above. You must submit your program for each test case separately, as Problem 2 (A), Problem 2 (B), …, Problem 2 (H). If your program succeeds for all 8 cases, you have succeeded in solving this programming question.

SOEN 6501 **PCT SESSION #2** **Mar 18, 2017 (10:00 – 12:30hrs)**

Student Id: **Name:** **Signature:**

Login to the PC² Lab System. Refer to the *SOEN6501LabInstructionForStudents* document for instructions. You can also download the version available on the *H:* drive and save it on your desktop.

Reminder:

When submitting your solution to the lab system, make sure there is no `package` statement at the top of your `.java` file as this will result in a grade of 0 (restriction of this system).

Unique Word Count (Basic Problem)

Problem Statement:

Write a program to count the number of unique words in the input line. The program should read in one line of input text (could be empty) and output the number of unique words in that line. A word is defined as a sequence (of any length) of alphanumeric characters terminated by a space or by a newline. Word equivalence is not case sensitive. You may assume that the maximum number of characters in an input line is 2000. Further, assume that there will not be any characters other than alphanumeric (a-z, A-Z, 0-9) and white spaces (blank, tabs and newlines) in the input.

Please note that your program MUST produce the CORRECT results for all test cases within PC2. These test cases are generated internally and are not shown to you.

Input Specification:

One line of text as specified above.

Output Specification:

One integer with no leading zeroes or spaces and terminated by newline with no spaces before the newline.

Sample Input Output:

Input:
Prithviraj Dajisaheb Chavan
Output:
3

Input:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
Output:
1

Input:
Prithviraj D Chavan Output chavan PrithviraJ d Input   Barack Obama InputOutput Output
Output:
8

Note: There are 8 test cases named A to H for this problem. The first three, A, B and C are revealed to you above. You must submit your program for each test case separately, as Problem 1 (A), Problem 1 (B), …, Problem 1 (H). If your program works correctly for all 8 cases, you have succeeded in solving this programming question.

Reminder:

When submitting your solution to the lab system, make sure there is no `package` statement at the top of your `.java` file as this will result in a grade of 0 (restriction of this system).

Game of Life (Advanced Problem)

Problem Statement:

The game of life is a game which is played on a field of cells. Cells are arranged in a rectangular grid. Each cell has at most eight neighbours, i.e. adjacent cells. Each cell in the field is either occupied by an organism or is blank.

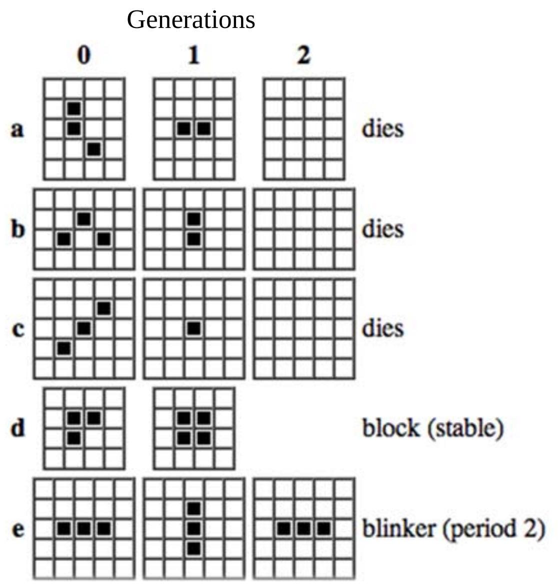
The objective of the game is to simulate the life of the organisms present in the field, generation after generation. The only work of the organism is to either reproduce or die!

The rules of reproduction are given as follows (examples in figure):

1. If an organism has 0 or 1 neighbours, it will die out of loneliness in the next generation
2. If an organism has 4 or more neighbours, it will die of crowding in the next generation
3. If an organism has 2 or 3 neighbours, it will prosper and survive into the next generation
4. If an unoccupied cell has exactly three neighbours, then an organism takes birth in the next generation

You are required to write a Java program which simulates the life of the organisms and outputs the state of the field after 'N' generations.

Please note that your program MUST produce the CORRECT results for all test cases within PC2. These test cases are generated internally and are not shown to you.



Source: Wikipedia

Input Specification:

The first line of input will be two integers, R and C, separated by a space, specifying the number of rows and columns that the field has. Assume that the maximum number of rows and columns will never be greater than 30. The next R lines of input contain C characters, consisting of '#' and '@' respectively. '#' means a blank cell, while '@' indicates an organism. The last line of input contains one integer N, which is the number of generations that are supposed to be simulated.

Output Specification:

The output of your program should be a single integer indicating the number of living organisms at the end of simulation, i.e., after N generations. Sample input and output for 2 examples is given below

Input:

```
3 3
#@#
#@@
@#@
3
```

Output:

```
3
```

Input:

```
3 3
#@#
#@@
@#@
4
```

Output:

```
4
```

Note: There are 7 test cases named A to G for this problem. The first two, A and B are revealed to you above. You must submit your program for each test case separately, as Problem 2 (A), Problem 2 (B), …, Problem 2 (G). If your program works correctly for all 7 cases, you have succeeded in solving this programming question.

Student Id: **Name:** **Signature:**

Login to the PC^2 Lab System. Refer to the *SOEN6501LabInstructionForStudents* document for instructions. You can also download the version available on the *H:* drive and save it on your desktop.

Reminder:

When submitting your solution to the lab system, make sure there is no `package` statement at the top of your `.java` file as this will result in a grade of 0 (restriction of this system).

Horizontal Vertical Intersections (Basic Problem)

Problem Statement:

Archeologists have discovered a very ancient cave, with many pictures on its wall. Each picture contains only vertical and horizontal line segments. To extract the meaning from these glyphs, it is necessary to find out how many of these vertical and horizontal line segments intersect. Each picture glyph is inscribed in a square of side 100. A vertical line segment is described by an X value and two Y values (X Y1 Y2). Similarly, a horizontal line segment is described by a Y value and a pair of X values (Y X1 X2). All the X and Y values are integers in the range 0 to 100. For example, a horizontal line segment described by a triplet (12 8 57) is a segment joining points (8, 12) and (57, 12).

Write a Java program to compute and output the total number of intersections in a picture glyph, given the triplets for line segments in that picture glyph. Only intersections between a horizontal and vertical line segments are counted. If one segment touches another, it is counted as an intersection. Assume that the maximum number of line segments is 200.

Please note that your program MUST produce the CORRECT results for all test cases within PC2. These test cases are generated internally and are not shown to you.

Input Specification:

The first line has two integers, V and H. V denotes the number of vertical line segments and H denotes the number of horizontal line segments in the picture. The next V lines contain triplets of integers (X Y1 Y2) for the vertical segments., and the next H lines describe H line contain triplets of integers (Y X1 X2). All numbers are separated by one or more spaces.

Output Specification:

One integer (number of intersections) with no leading zeroes or spaces and terminated by newline with no spaces before the newline.

Sample Input Output:

Problem 1 (A)

Input:

```
1 1
50 20 60
30 40 80
```

Output:

```
1
```

Problem 1 (B)

Input:

```
2 2
25 25 80
75 25 80
25 25 80
75 25 80
```

Output:

```
4
```

Problem 1 (C)

Input:

```
1 2
50 0 100
50 0 75
50 25 100
```

Output:

```
2
```

Note: There are 8 test cases named A to H for this problem. The first three, A, B and C are revealed to you above. You must submit your program for each test case separately, as Problem 1 (A), Problem 1 (B), …, Problem 1 (H). If your program works correctly for all 8 cases, you have succeeded in solving this programming question.

Reminder:

When submitting your solution to the lab system, make sure there is no `package` statement at the top of your `.java` file as this will result in a grade of 0 (restriction of this system).

Hashing (Advanced Problem)

Problem Statement:

Consider a hash table of size N, numbered 0 to N-1. You have to insert integers into this table using the hashing technique given below:

Let ***i*** be the integer to be inserted. Compute the index ***j*** of the location where the insertion is to be made as *j = i mod N*. If this location is empty then put the element at this position else recompute the next location as follows:

Remove the right most digit of ***i*** and recompute ***j = i mod N***. If the digit removed was odd, then move ***j*** locations forward from the current location else move ***j*** locations backward from the current location (assume 0 as even). Note that this move will wrap around both the edges of the table.

Keep doing this till you either find a free location or all the digits of ***i*** have been removed. If all digits of ***i*** have been removed and yet unable to find a free location, then from the last location tried, start moving in the direction corresponding to the last digit removed. Keep moving till you detect a free location.

Assume that the number of integers inserted is not more than the table size.

Input Specification:

The first line will contain just one integer. This will give the table size, N. On the next line will be the list of positive integers that need to be inserted into the table. The integers will be separated by one or more spaces, and the last integer will be -1 indicating end of input. (-1 is not to be inserted into the table).

Output Specification:

The output should contain, for each integer, the locations that were checked while inserting that integer (including the location in which the integer was finally inserted). The locations checked for each of the integers should be output on a line by itself, separated by one space each, each line being terminated by a new line with no spaces before the newline.

Sample Input Output:

Problem 2 (A)

Input:

```
7
38 52 145 16 179 4 -1
```

Output:

```
3
3 5
5 5 4
2
4 0
4 4 3 2 1
```

Note: There are 6 test cases named A to F for this problem. The first one, A, is revealed to you above. You must submit your program for each test case separately, as Problem 2 (A), Problem 2 (B), …, Problem 2 (F). If your program works correctly for all 6 cases, you have succeeded in solving this programming question.

SOEN 6501	PCT SESSION #4	June 17, 2017 (10:00 – 12:30hrs)
Student Id:	Name:	Signature:

Login to the PC² Lab System. Refer to the *SOEN6501LabInstructionForStudents* document for instructions. You can also download the version available on the *H:* drive and save it on your desktop.

Reminder:

When submitting your solution to the lab system, make sure there is no `package` statement at the top of your `.java` file as this will result in a grade of 0 (restriction of this system).

Bracket Matching (Basic Problem)

Brackets are punctuation marks used in matched pairs within text, to set apart or interject other text. There are different types of brackets, parantheses, square, curly, etc. Write a program which reads a String, which consists of alphabets [a-z, A-Z, space] and 3 types of brackets shown below:

Parentheses - () Square brackets - [] Braces or Curly brackets - {}

And determine whether every open bracket has a matching close bracket and is properly nested. If any open/close bracket doesn't have a matching close/open bracket or any extra open/close bracket then it is to be treated as invalid string.

Following are 4 examples of valid string:	Following are 4 examples of invalid string:
(the[is]{valid})	(the[is]{invalid))
{the(is[valid])}	(the[is]{invalid}}
(this)(is)(valid)	(this](is}{invalid))
([{this is valid}])	([{this is invalid]})

Please note that your program MUST produce the CORRECT results for all test cases within PC2. These test cases are generated internally and are not shown to you.

Input Specification:

First line of the input consists of an integer N, followed by N number of strings with each string is on a separate line.

Output Specification:

For each input string, print "TRUE" if it is a valid string else print "FALSE", terminated by newline character.

Note: Please do not include any extra text output in your program, like "Enter number of strings" etc. Your program is checked automatically and it must produce the exact output as expected above, no blank lines, no punctuations, or other superfluous characters.

Sample Input Output:

Problem 1 (A)

Input:

```
4
(the[is]{valid})
(the[is]{valid))
{the(is[valid])}
(this](is}{valid)
```

Output:

```
TRUE
FALSE
TRUE
FALSE
```

Note: There are 6 test cases named A to F for this problem. The first one, A, isrevealed to you above. You must submit your program for each test case separately, as Problem 1 (A), Problem 1 (B), …, Problem 1 (F). If your program works correctly for all 6 cases, you have succeeded in solving this programming question.

Reminder:

When submitting your solution to the lab system, make sure there is no package statement at the top of your .java file as this will result in a grade of 0 (restriction of this system).

Deadlock Detection (Advanced Problem)

Deadlock detection is an important consideration in operating systems of computers. In this problem you will be given a directed graph representing the state of a system. Your program should detect whether there are any deadlocks in the system. A deadlock can be detected by checking for cycles in the graph. The input will be in the form
Process X holds resource Ri and wants resource Rj (X holds Ri means an edge Ri->X and X wants Rj means an edge X->Rj).
Note that both processes and resources form nodes of the graph.

Use depth first traversal to determine whether there are any cycles in the graph. The graph contains a cycle if while expanding a node, you encounter a child that is present in the path from the root of the current tree to the node being expanded. You can assume that the maximum number of nodes in the graph will not exceed 50.

Input Specification:
An integer N and a sequence of N lines each containing 3 integers.
The first integer represents the process id X, the second represents the resource id Ri that the process X is holding and the third represents the resource id Rj that the process X wants. A resource id of -1 should be ignored.
A process can hold any number of resources and can ask for any number of resources. These are given one per line. For example, in the sample input below, process 101 holds 7 and 9 and wants 8.
The integers denoting the process id and those denoting resource id's are known to be disjoint.

Output Specification:
The program should print YES if it detects a deadlock and NO otherwise. Terminate output with a newline.

Note: Please do not include any extra text output in your program, like "Enter number of lines" etc. Your program is checked automatically and it must produce the exact output as expected above, no blank lines, no punctuations, or other superfluous characters.

Sample Input Output:
Input

```
5
101 7 8
101 9 -1
102 -1 7
102 8 -1
103 -1 9
```

Output

```
YES
```

Note: There are 6 test cases named A to F for this problem. The first one, A, is revealed to you above. You must submit your program for each test case separately, as Problem 2 (A), Problem 2 (B), …, Problem 2 (F). If your program works correctly for all 6 cases, you have succeeded in solving this programming question.