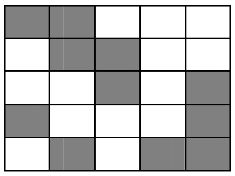
Practical Test (16. 6. 2016)



Problem 1: Consider a two-dimensional grid of cells, each of which may be empty or filled. Filled cells form ‘Gray’. The filled cells that are connected form the same bigger Gray. Two cells are said to be connected if they are adjacent to each other horizontally, vertically, or diagonally, and also these two cells are Grays. There may be several Grays on the grid. Your job is to find the smallest Gray (in terms of number of cells) on the grid. Write a program that determines the size of the largest Gray for a given set of Grays.

Input: The input begins with a single positive integer ‘n’ on a line by itself indicating the number of rows and columns. And the grid is given as a set of strings, each composed of 0’s and 1’s. The ‘1’ indicates that the cell is filled and ‘0’ indicates an empty cell. The strings should be converted into the grid format.

Output: The output is the size of the largest Gray found on the grid.

Input	**Output**
5 11000 01100 00101 10001 01011	5

===

Problem 2: When you first made the computer to print the sentence “Hello World”, you felt so happy, not knowing how complex and interesting the world of programming and algorighm will turn out to be.

#include<iostream.h>
void main(){
cout<<”Hello World!”;
}
	#include<iostream.h>
	void main(){
	cout<<”Hello World!”;
	cout<<”Hello World!”;
	}
		#include<iostream.h>
		void main(){
		cout<<”Hello World!”;
		cout<<”Hello World!”;
		cout<<”Hello World!”;
		cout<<”Hello World!”;
		}
			#include<iostream.h>
			void main(){
			cout<<”Hello World!”;
			cout<<”Hello World!”;
			cout<<”Hello World!”;
			cout<<”Hello World!”;
			cout<<”Hello World!”;
			cout<<”Hello World!”;
			cout<<”Hello World!”;
			}

Figure 1	**Figure 2**	**Figure 3**	**Figure 4**

Then you did not know anything about loops, so to print 7 lines of “Hello World!”, you just had to copy and paste some lines. If you were intelligent enough, you could make a code that prints “Hello World!”, 7 times, using just 3 paste commands. Note that we are not interested about the number of copy commands required. A simple program that prints “Hello World!”, is shown in the Figure. By copying that single print statement and pasting it we get a program that prints two “Hello World!” lines. Then copying these two print statements and pasting them, we get a program that prints four “Hello World!” lines. Then copying three of these four statements and pasting them we can get a program that prints seven “Hello World!” lines. So three pastes commands are needed in total and of course you are not allowed to delete any line after pasting. Given the number of “Hello World!” lines you need to print, you will have to find out the minimum number of pastes required to make that program from the origin program shown in Figure 1.

Input: the input contains up to 200 lines of inputs. Each line contains an integer N that denotes the number of “Hello World!” lines are required to be printed. Input is terminated by a line containing a negative integer.

Output: For each line of input except the last one, produce one line of output of the form ‘Case<1 space>X:<1 space>Y’ where X is the serial of output and Y denotes the minimum number of paste commands required to make a program that prints N lines of “Hello World!”.

Input	**Output**
2 10 -1	Case 1: 1 Case 2: 4

===

Problem 3: You have devised a new encryption technique which encodes a message by inserting between its characters randomly generated strings in a clever way. Because of pending patent issues we will not discuss in detail how the strings are generated and inserted into the original message. To validate your method, however, it is necessary to write a program that checks if the message is really encoded in the final string. Given two strings s and t, you have to decide whether s is a subsequence of t, i.e, if you can remove characters from t such that the concatenation of the remaining characters is s.

Input: The number in the first line indicates the number of test cases. Each is specified by two strings s, t of alphanumeric ASCII characters separated by whitespace.

Output: For each test case, output 1 if s is a subsequence of t.

Input	Output
4	
sequence subsequence	1
person compression	0
VERDI vivaVittorioEmanueleReDiItalia	1
caseDoesMatter CaseDoesMatter	0

===

Problem 4: The ISSN (International Standard Serial Number) is an eight-digit number which identifies periodical publications as such, including electronic serials. Each ISSN is unique to a journal publication. Samples of ISSN from taken from various electronic journals are shown below:

ISSN 1144-875X

The Open Electrical & Electronic Engineering Journal

ISSN: 1874-1290

The ISSN takes the form of the acronym ISSN followed by two groups of four digits, separated by a hyphen. The eighth character is a control digit calculated according to a modulo-11 algorithm on the basis of the 7 preceding digits; this eighth control character may be an X if the result of the computing is equal to 10, in order to avoid any ambiguity. The steps for calculating the control digit of ISSN is as follows:

- Take the first seven digits of the ISSN.
- Multiply each digit in turn by its weighting factor: 8, 7, 6, 5, 4, 3, 2.
- Add these numbers together.
- Divide this sum by the modulus 11.
- If the remainder is zero, it can be used as the control digit.
- Otherwise, subtract it from 11 to obtain the control digit.
- If this value is 10, substitute an upper case 'X' in the control digit position.

Write a program to take in the two groups of digits separately without the last control digit and display the complete ISSN number (including the control digit) to user.

Sample Input	Sample Output
3	0317-8471
0317-847	1874-1290
1874-129	1144-875X
1144-875	

Input: The first line of the input contains an integer N (1 <= N <= 10) denoting the number of test cases. Each test case contains a first seven-digits of the ISSN.

Output: For each test case, your program should output the eight-digits ISSN.

Problem 5: The movie ticket pricing to a regular cinema hall is shown below.

Period / Price	Standard Ticket Price	Senior Citizen (show time before 1800hrs)	Student (show time before 1800hrs)
Off-Peak (Monday to Thursday)	$8.50	$4.00	$6.50
Peak (Friday to Sunday, Eve of Public Holiday, Public Holiday)	$11.00		

Using the information given above, develop a program to calculate the total cost of a family going for a movie. Your program should output the total cost.

Input: The first line of the input represents the number of the adult, the second line represents the number of the senior, the third line represents the number of the student, the fourth line represents the day number, the fifth line represents the public holiday or not and the last line represents the show time before or after 1800 hrs.
Use 1 for Monday, 2 for Tuesday, 3 for Wednesday, 4 for Thursday, 5 for Friday, 6 for Saturday, 7 for Sunday.
If the day is peak day, represent it as 1. Otherwise, use it as 0.
Use 1 for the show time before 1800hrs, otherwise, use 0.

Output: The program output is the total cost of the family with $ sign.

Explanation: 2 means two adults, 1 means one senior, 1 means one student, 4 means Thursday, 0 means off-peak, 1 means show time before 1800hrs.

Sample Input	Sample Output
2	$27.5
1	
1	
4	
0	
1	

===