ACM International Collegiate Programming Contest (ICPC) Asian Region Contest, Hong Kong, 2000 Problem Set

October 28, 2000

Question 1: Multiple Exponentiation

Modulo arithmetic is an interesting topic as there are different ways of computing the results effectively. Consider the multiple exponentiation problem, which computes the following modulo arithmetic:

$$y = a^{p^{p^{\cdot^{\cdot^{\cdot^{p}}}}}} \; mod \; n$$

where there are m exponentiations. The numbers a, p, m and n are all positive integers greater than 1 and less than 65535. Furthermore, a and p are prime numbers such that $n \leq 2a$ and $n \leq 2p$. This ensures the existence of useful discrete logarithms.

Write a program to compute the value of y. Your program should be structured as a loop to read in lines of 4 numbers, in the order of a, p, m and n, from a file "q1.txt" and print out the results to a file "out1.txt". Submit your source program together with either an executable file called "q1.exe" (for c) or a class file called "q1.class" (for Java). In your program, no error checking is required for input. Sample inputs and outputs of the program are as follows:

Input:

```
3 3 1 5
3 3 2 5
3 3 3 5
47 47 1 67
47 47 2 67
47 47 3 67
32719 54323 99 65399
```

Output:

```
2
2
2
6
19
54
46184
```

Question 2: Complex Arithmetic

Complex numbers can be useful in many context. Here we would like to perform complex arithmetic and evaluate an expression containing complex numbers. Each simple complex number is of the form $a+bi$ or $a-bi$, or simply bi, where a and b are positive integers. Thus $-a+bi$ and $-a-bi$ are illegal forms. There are five operators in complex expressions: "+", "-", "*", "/" and "%", where "%" is a unary postfix operator meaning conjugate. In other words, $(a+bi)\% = a-bi$ and $(a-bi)\% = a+bi$. There is no unary minus operator.

In this problem, the precedence rules of the operators are *slightly different* from those in common sense. In particular, the conjugate operator "%" will take the highest precedence. Both "+" and "-" will take the next precedence. Finally, "*" and "/" will take the lowest precedence. Operators with the same precedence are evaluated in a left-to-right order. In other words, subexpressions within brackets will be evaluated first. The conjugate will be evaluated before "+" and "-", which again will be evaluated before "*" and "/". Each expression will be terminated by the "=" operator that will force the expression to be evaluated.

Operands to the expressions can be either a positive number or a simple complex number, but not "complicated" complex numbers. It is possible that during evaluation of division, complex numbers that are not simple may be resulted. In this application, you are required to express "complicated" complex numbers in the form of $[a+bi]/r$ or $[a-bi]/r$, where a and b are non-negative integers and r is a non-zero integer (may be negative). Note that $(a+bi)/(c+di) = [(ac+bd)+(bc-ad)i]/(c^2+d^2)$.

Write a program to read in a sequence of lines from an input file containing one or more expressions, each ended with "=". For each expression, evaluate and print the result in the form of a "complicated" complex number, $[a+bi]/r$ or $[a-bi]/r$. The input file is named "q2.txt". The result is to be printed to an output file "out2.txt". Submit your source program with either an executable file called "q2.exe" (for c) or a class file called "q2.class" (for Java). In your program, no error checking is required for input. For simplicity, each line can be assumed to contain at most 80 characters and that during all intermediate steps, the numerator and denominator will not fall beyond the range of a largest signed integer. Sample inputs and outputs of the program are as follows:

Input:

```
3+2i * 4 + i =
2+i*
2-i =
(2 + 3 i) /(7+i)% =
(5+3i)/(1+3i)% =
(1+i)%/3+ i *2-i+3 =
((1+i)%/3+i*2-i+3)+(3i+1*(2i*2-i%/3)) =
```

Output:

```
[10+11i]/1
[5+0i]/1
[11+23i]/50
[2-9i]/-5
[3-11i]/5
[61+43i]/-15
```

Question 3: The Moving Problem

Johnny is the manager of a moving company. As business is improving by the days and Johnny would like to optimize the operations and profits of the company, he decides to automate manpower planning and scheduling. The main factors to consider in such a planning exercise include what major furniture items are to be moved and the manpower requirement (such as number of movers) to move each of such items. Johnny's customers are usually corporate clients who are demanding and always like to have the moving job completed within a certain amount of time. Johnny comes to you for help for such a planning and scheduling system.

Write a program to read in a sequence of job descriptions from a file named "q3.txt". Each job description consists of:

- the maximum time allowed for moving all items,
- the number of major furniture items, followed by
- a list of records, one for each of the major furniture items to move and each on a separate line.

Each record contains three fields:

- the name (at most 30 characters long) of the item,
- how long it takes to move the item, and
- the number of movers required for moving the item.

Time is specified in minutes.

Each job description should generate the following output:

- A line printing "Job description:".
- The job description:
  - One line is output for each major furniture item, its time requirement, and manpower requirement, separated by commas. Each line is enclosed by a pair of square brackets.
  - The sequence of furniture items should be ordered in exactly the same way that they are read in.
- A line with "Number of movers required: " followed by the minimum number of movers required to satisfy all the timing and manpower requirement constraints.
- All possible schedules for moving each of the major furniture items, specifying the start times for each item.

Each schedule consists of a sequence of start times on one line. The times should be enclosed in square brackets and separated by commas. Each start time is for each furniture item. The start times should be printed in the exact order that the items are read in. The time should be in the number of minutes since the start of the moving job (at time 0).

Note that the many possible schedules of a job must be ordered ascendingly according to the lexicographical ordering defined as follows. We consider a schedule for n items to be a string of n integers

$$t_1, t_2, ..., t_n.$$

A string $t_1, ..., t_n$ is *less than* another string $s_1, ..., s_n$ if there exists $1 \leq k \leq n$ such that

- $t_i = s_i$ for all $i < k$, and
- $t_k < s_k$.

Output for different jobs should be separated by a blank line. The job output should also be ordered in exactly the same way that they are specified in the input file. The output should be printed to the file named "out3.txt".

You can safely assume that all items can be moved at the same time if enough manpower is assigned. You also need to consider only jobs with at most ten furniture items. Each job spans across at most one hour. Each furniture item requires at least one mover.

Example input:

```
60
4
piano 30 3
chair 10 1
bed 15 3
table 15 2
60
4
piano 30 3
chair 10 1
bed 15 3
table 15 3
```

The above sample input specifies two jobs. The first job must be completed within an hour and consists of four major furniture pieces. The first piece is a piano requiring thirty minutes and three movers to move. The other items can be interpreted similarly. The second specification begins immediately after the first specification. The second job is very similar to the first job except for a slight difference in manpower requirement for the table.

Corresponding Output:

```
Job description:
[piano,30,3]
[chair,10,1]
[bed,15,3]
[table,15,2]
Number of movers required: 3
[0,30,45,30]
[0,31,45,30]
```

```
...
[30,19,0,15]
[30,20,0,15]

Job description:
[piano,30,3]
[chair,10,1]
[bed,15,3]
[table,15,3]
Number of movers required: 4
[0,0,30,45]
[0,0,45,30]
[0,1,30,45]
...
[30,49,15,0]
[30,50,0,15]
[30,50,15,0]
```

The output format up to the schedules for each job is pretty straightforward. For space reason, we cannot give you all the possible schedules, but they must respect the same lexicographical ordering defined above. For example, in the first schedule of the first job, it means that the piano should be handled at the beginning of the move, the chair and the table both after 30 minutes, and the bed after 45 minutes. The first schedule of the first job is lexicographically less than the second schedule of the same job since

- they are equal in the first start time, and
- the second start time (30) of the first schedule is less than that (31) of the second schedule.

This lexicographical ordering is maintained for all schedules of both the first and second jobs. Note that a blank line is printed between the output of the first job and that of the second job.

You can safely assume that there are at least one item to move in each moving job, and that all job specifications are doable when given enough manpower resources. Your program should also run with reasonable efficiency. We expect your program to finish our test cases within around five minutes.

Submit your program executable under the name “q3.exe” (for c) or “q3.class” (for Java).

Question 4: Network Watcher

In order to guarantee the effectiveness of a communication network, we will monitor each communication line to make sure it works properly. We will use a device, called *Network Watcher*, for this purpose. Once a Network Watcher is installed on a site, it can monitor all communication lines connected to the site. To be cost effective, we wish to use the smallest number of Network Watchers to monitor all communication lines in a communication network. As you may have guessed, it is extremely difficult to determine such a number for a network.

Fortunately, the problem you are facing seems easier: there are only K Network Watchers available. Therefore if a network requires more than K Network Watchers, we have to use other means to monitor the communication lines. Your task is to determine if a given network requires at most K Network Watchers to monitor all communication lines; if so, find at most K sites to install Network Watchers to monitor all communication lines in the network. Call this the K Network Watcher problem. The really good news for you is that due to budget constraints, K is at most 10. Furthermore, the number N of sites in a network is at most 100.

Write a program to solve the K Network Watcher problem. You program should be efficient enough to solve the problem instances in time (around five minutes). Your program should read the input from a file named "q4.txt". The first line of the input is the number of networks in the input. Assume that each site is represented by a unique number between 1 and N. The input for each network consists of numbers N and K, and a description of the network. The first line of input contains numbers N and K separated by exactly one blank space. It is followed by at most N lines that specify a network, and number -1 in a separate line. For site with site index i (or simply site i), there is an input line consisting of the site followed by the list of sites with larger site indices that are connected to site i, with a blank space separating every two sites. For example, the following line

1 4 2 3 6 7

means that site 1 is connected to sites 2 3 4 6 and 7. If site i is not connected to any sites with larger site indices, there is no input line for site i.

Your output should be printed to a file named "out4.txt". The output for each problem is a list of sites, each separated by one blank space, and should be on a separate line. The list of sites should be ordered in increasing site indices. If K Network Watchers are not enough for an input network, you just output "Impossible".

Example Input:

```
2

8 3
1 4 2 3 6 7
2 3 4 5 6
3 5 7 6 8
-1

6 3
1 4 5 2
2 6 3
3 4 6
```

```
4 5
5 6
-1
```

Corresponding Output:

```
1 2 3

Impossible
```

You should note that the solution to a K Network Watcher problem is not unique. However, you are required to give only one answer.

Submit your program executable under the name "q4.exe" (for c) or "q4.class" (for Java).

Question 5: Tiling a Board

Consider an $n \times n$ ($17 \geq n \geq 2$) board of unit squares with the upper-right corner square missing. Figure 1 shows some of the possible boards for $n = 2$, 3, and 4. The missing squares are shaded.

You are required to tile a board using triominoes (Figure 2). In this tiling two triominoes may not overlap and they must cover all the $(n^2 - 1)$ squares.

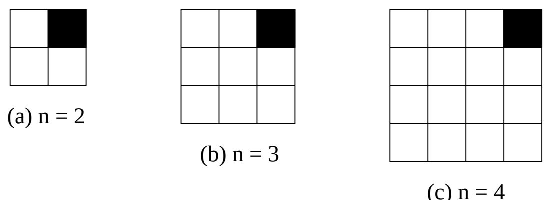

Figure 1: Boards

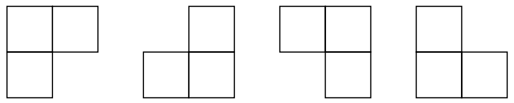

Figure 2: Triominoes with different orientations

Write a program that shows how a board could be tiled by triominoes. Your program should read an input file "q5.txt" that contains a few lines. Each line contains an integer n. If an n by n board with a missing corner square can be tiled, your program should output the tiling by marking every square with a tile number. Specifically, for each input number n, your program should output n lines, each line with n integers separated by a space (except for the first line, which has $n - 1$ integers). The i-th number of the j-th line represents the number of the tile that covers the square in the board that is i units from the left and j units from the top. Format your output so that each integer occupies exactly two character spaces.

If the board cannot be tiled, your program should output "No solution". Write your output to a file named "out5.txt".

Submit your program executable under the file name "q5.exe" (for c) or "q5.class" (for Java).

Example input:

```
3
4
5
10
```

Corresponding output:

```
No solution

 2  2  5
 2  3  5  5
 1  3  3  4
 1  1  4  4

 3  3  6  6
 3  5  5  6  8
 2  2  5  8  8
 2  1  4  4  7
 1  1  4  7  7

17 17 19 19 21 21 29 29 33
17 18 19 20 21 22 29 32 33 33
18 18 20 20 22 22 30 32 32 31
 1  1  3  3  6  6 30 30 31 31
 1  2  3  5  5  6  7 28 28 27
 2  2  4  4  5  7  7 28 27 27
16 16 12  4  8  8  9 26 26 25
16 13 12 12  8  9  9 26 25 25
15 13 13 14 10 10 11 24 24 23
15 15 14 14 10 11 11 24 23 23
```

Question 6: Switch Toggling

Consider an n by n board of n^2 squares. Each square has a pair of co-ordinates (x,y) where $1 \leq x, y \leq n$. Each square has two states: either "On" or "Off". Initially, all squares are in the "Off" state.

For a square (i,j), its *neighbors* are the squares: $\{(i', j')|1 \leq i', j' \leq n; |i' - i|, |j' - j| \leq 1\}$. For example, if $n = 4$, the neighbors of (4,2) are $\{(4, 1), (4, 2), (3, 2), (4, 3)\}$. Note that ($i$,$j$) is a neighbor of itself.

Define the operation *toggle*(i, j) that toggles the states of square (i, j)'s neighbors. That is, if a square's original state is "On", it will be switched to "Off" and vice versa.

Write a program that, given an n by n board of all "Off" squares, computes a *shortest* sequence of *toggle* operations that turns all the squares "On". If no sequences can turn all the squares "On", output "No solution".

Specifically, your program should read an input file named "q6.txt" that contains a few lines. Each line contains a number n. Your program should output a file named "out6.txt", which contains n sequences (or the string "No solution"), separated by blank lines. The i-th sequence gives a solution to the toggling problem of the board specified by the i-th input number n. Each sequence is a sequence of square co-ordinates of the form (a,b). (Note that a and b are separated by a lone comma, not any spaces.) The co-ordinates of squares are separated by a space. A sequence specifies the order in which the *toggle* operation should be applied to the squares. For example, the sequence:

$$(1, 2)\ (2, 3)\ (1, 1)$$

specifies that the toggle operations are applied to square (1,2) first, then to square (2,3), and finally to square (1,1). Figure 3 shows how the above sequence changes the states of the squares in a 3 by 3 board. (Shaded squares are "On".)

Submit your program executable under the file name "q6.exe" (for c) or "q6.class" (for Java).

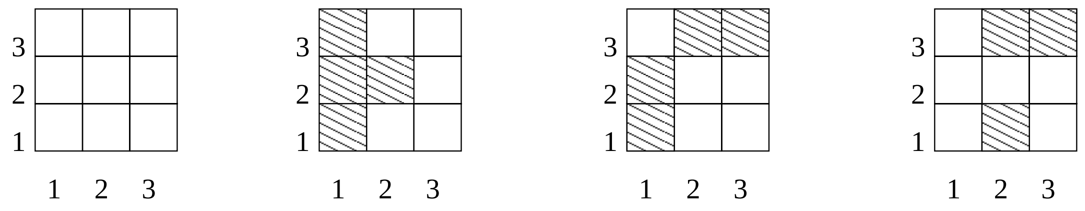

Figure 3: Boards

Example input:

```
2
3
4
```

Corresponding output:

```
(1,1) (1,2) (2,2) (2,1)

(2,2) (1,3) (3,3) (3,1) (1,1)

(1,1) (1,3) (1,4) (2,2) (2,3) (2,4) (3,1) (3,2) (3,3) (4,1) (4,2) (4,4)
```