Programming Design, Spring 2014
Lab Exam 2

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

Submission and grading. The exam starts at 5:30 pm on May 15 and ends at 10:30 pm on May 15. This is a take-home exam, so you may work at any place you like, as long as you have Internet access to submit your work. TAs will stay in the computer classroom (the one we did lab exam 1) and online between 6:30 and 8:30. If you need clarifications for problems, you may find them in the classroom or send them e-mails. If your e-mails are sent by 8:30, we promise to give you a response in time; for e-mails after 8:30, no guarantee. Facebook messages may or may not be replied; it is up to the TAs.

In this exam, there are three problems. You need to write a C++ program for each problem. 100% of your grades will be based on the correctness of your outputs. The online grading system will input in total 50 lines of testing data and then check your outputs. These 50 lines count for 100 points, i.e., 2 points for each set. Before the due time of the exam, you may upload your programs multiple times. Only the last one you upload will be graded. Unlike what happens with your homework, you will not see your scores during the exam. To submit your work, please upload the three .cpp files, one for each problem, to the online grading system at http://lckung.im.ntu.edu.tw/PD/.

Discussing the exam with anyone during the exam period, directly or indirectly, is definitely cheating.

Problem 1

(30 points) Given an $n \times n$ matrix A, its trace is

$$\sum_{i=1}^{n} A_{ii},$$

i.e., the sum of diagonal elements. At the same time, its maximum element is

$$\max_{i=1,...,n,j=1,...,n} A_{ij}.$$

Obviously, for some matrices the trace is greater while for some other matrices the maximum element is larger. In this problem, you need to calculate the two values for a given matrix and determine which one is larger.

Input/output formats

The input consists of 15 lines. In each line, there are $n^2 + 1$ integers n, A_{11}, A_{12}, ..., A_{1n}, A_{21}, ..., and A_{nn}, where $0 < n \leq 100$ and $-100 \leq A_{ij} \leq 100$ for all $i = 1, ..., n$, $j = 1, ..., n$. Two consecutive integers are separated by one white space. Your program should read these integers and then output the following:

- If the trace is strictly larger, output a letter "T", then the trace, then the maximum element.
- If the maximum element is strictly larger, output a letter "M", then the maximum element, then the trace, then i and then j where the maximum element locates at row i and column j. If multiple elements are all the maximum, only output the i and j for the one with the largest $n(i-1)+j$.
- If the trace and maximum element are equal, output a letter "E" and then one of them.

Two consecutive integers in the output should be separated by a white space. After the last integer there should be a new line character. For example, suppose we have

```
1 1
2 2 0 -1 4
3 1 2 4 4 1 2 2 4 1
```

as a line of input, the output should be

```
E 1
T 6 4
M 4 3 3 2
```

with a new line character appended at the end.

Problem 2

(20 points) For the same task and output requirement as Problem 1, now the input format is changed: The number n is not given to you. The input consists of 10 lines. In each line, there are n^2 integers A_{11}, A_{12}, ..., A_{1n}, A_{21}, ..., and A_{nn}, where $-100 \leq A_{ij} \leq 100$ for all $i = 1, ..., n$, $j = 1, ..., n$. Two consecutive integers are separated by one white space. For example, suppose we have

```
1
2 0 1 4
1 2 4 4 1 2 2 4 1
```

as three lines of input, the output should be

```
E 1
T 6 4
M 4 3 3 2
```

with a new line character appended at the end.

Problem 3

(50 points) Recall that we have studied a scheduling problem for minimizing makespan over identical machines. In this problem, we will work with parallel machines, i.e., different machines have different production rates. In a given instance, let m and n be the number of machines and jobs, respectively. For job j, the workload q_j is interpreted as the number of products to be produced. For machine i, its production rate p_j determines the number of products that can be made by it in a unit time. Therefore, if we assign job j to machine i, the processing time required for completing job j is $\frac{q_j}{p_i}$. In other words, assigning a job to different machines may result in different processing time.

Obviously, the previous problem with identical machines is a special case of the problem with parallel machines (when all the production rates are identical). It is thus conceivable that the problem with parallel machines is even harder than that with identical machines. Therefore, finding an optimal schedule that minimizes makespan is believed to be too time-consuming in general.

Anyway, in this problem you are asked to implement the following algorithm only. First, sort jobs by their required quantity q_j, from large to small. Then assign jobs one by one following this order. When we assign a job, assign it to a machine that results in the smallest makespan up to this job. For example, suppose there are ten jobs with required quantities 9, 9, 7, 7, 5, 5, 5, 4, 3, and 2 and three machines with production rates 1, 2, and 1, the assignment of jobs can be summarized in the following table. Note that at round 5, the job is assigned to machine 2 (which results in the smallest makespan: $8 + 2.5 < 7 + 5 < 9 + 5$), not machine 3 (the one that currently finishes the earliest).

$\frac{q_j}{p_i}$	Round										Total processing time
	1	2	3	4	5	6	7	8	9	10	
Machine 1		9						4			13
Machine 2	4.5			3.5	2.5		2.5		1.5		14.5
Machine 3			7			5				2	14

If two jobs have the same required quantity, the one with the smaller job ID goes first. If two machines both result in the same minimum makespan up to the job that is being scheduled, assign the job to the machine with the smaller machine ID (for the above example, this happens at round 2).

Input/output formats

The input consists of 25 lines. In each line, there are $n + m + 2$ integers m, n, p_1, ..., p_m, q_1, ..., q_n, where m is the number of machines, n is the number of jobs, p_i is the processing rate of machine i, and q_j is the required quantity of job j. It is given that $2 \leq m \leq 10$, $1 \leq n \leq 1000$, $m < n$, $q_i \leq 10000$ and is a multiple of 6, and $p_i \in \{1, 2, 3\}$. Two consecutive integers are separated by one white space. Your program should read these integers and then output the resulting makespan followed by the total processing times of all machines, from machine 1 to machine m. Two consecutive output numbers should be separated by a white space. After the last integer there should be a new line character. For example, suppose we have

```
3 8 2 1 1 30 30 24 24 24 18 18 12
2 8 1 1 18 12 12 6 6 6 6 6
```

as two lines of input, the output should be

```
48 48 42 42
36 36 36
```

with a new line character appended at the end.

To make it easier to earn points, the input will be organized as follows:

- For the first 10 lines, jobs will be sorted when they are given. In other words, in the input we have $q_1 \geq q_2 \geq \cdots \geq q_n$.
- For the 11th to 20th lines, the processing rate of all machines are 1. In other words, in the input we have $p_1 = \cdots = p_m = 1$.

Please note that in the first 10 lines, machines may have different processing rates; in the 11th to 20th lines, jobs are not sorted. Finally, in the last 5 lines, jobs are not sorted and machines have different processing rates.

Hint. If you do not know how to sort a set of jobs, at least you know how to find a job with the largest required quantity. In each iteration, do so, schedule it, and then delete it (there are many ways of doing it). This will still work this problem. Some sorting algorithms will be introduced after in the coming weeks.