2019 ECNA Regional Practice Contest

Practice 2019

October 25, 2019

ICPC North America Regionals 2019

icpc international collegiate programming contest

icpc global programming tools sponsor

icpc north america sponsor

ICPC East Central NA Regional Contest

Problems

A	Chopping on Line
B	Diamonds Are for Evers
C	Flight Turbulence
D	Hardware Sales
E	Make America Grade Again
F	This Problem's a Slam Dunk

Do not open before the contest has started.

International Collegiate Programming Contest
2019 East Central Regional PRACTICE Contest
Grand Valley State University
University of Cincinnati
University of Windsor
Youngstown State University
October 25, 2019

Rules:

1. There are **six** problems to be completed in **90 minutes**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. When displaying results, follow the format in the Sample Output for each problem. Unless otherwise stated, all whitespace in the Sample Output consists of exactly one blank character.
4. The allowed programming languages are C, C++, Java, Python 2 and Python 3.
5. All programs will be re-compiled prior to testing with the judges' data.
6. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contest officials (e.g., that might generate a security violation).
7. Programs will be run against multiple input files, each file containing a single test case.
8. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
9. All communication with the judges will be handled by the Kattis environment.
10. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A
Chopping on Line

Sawyer Steele owns a metal cutting facility. A customer will send him a description of a piece they need, Sawyer will trace the outline of the piece on a rectangular sheet of metal and then cut out the piece using straight-line cuts. Consider the current customer's order shown on the left in Figure A.1 where Sawyer has traced a triangle on a sheet of metal. Sawyer can set the machine to cut this piece using three cuts, as shown in the other three pictures: first along a line containing points A and B, then along a line containing points B and C, and finally along a line containing points A and C.

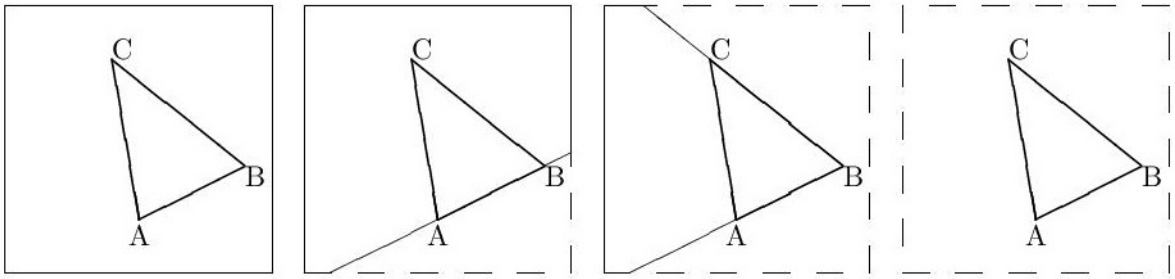

Figure A.1: Cutting example.

Sawyer is about to make these cuts when he starts wondering if these three cuts are the best ones to make. There are other ways to cut out this piece (say along the line containing A and C first, then the line containing C and B and then the line containing A and B). He's sure that each way that he could cut out this piece would result in a different total length of the cuts, and the smaller the length the less wear and tear on his machine.

Determining the minimum length of cuts is well beyond Sawyer's abilities so he's come to you for some help. Given the location of a triangle on a sheet of metal, he wants to know what cuts lead to the minimum total length of cuts. Do you think you have the coding ability to write such a program? Or in other words, do you have the mettle to cut it?

Input

Input consists of a single line containing eight non-negative integers w h A_x A_y B_x B_y C_x C_y, where w and h ($0 < w, h \leq 1\,000$) are the width and height of the rectangular sheet of metal (width goes along the x-axis, height along the y-axis) and the remaining six values are the x and y coordinates of the points A, B and C ($0 \leq A_x, B_x, C_x \leq w, 0 \leq A_y, B_y, C_y \leq h$). Points may lie on the edge of the rectangular sheet but no two points will ever lie on the same edge. The lower, left corner of the sheet of metal is at location $(0, 0)$ and the upper, right corner is at (w, h).

Output

Output the total length of all cuts for the optimal set of cuts, rounded to the nearest hundredth. On the next line indicate the order that the cuts should be made using the format p_1–p_2 p_3–p_4 p_5–p_6 where each p_i is

either `A`, `B` or `C`. The first pair p_1-p_2 lists the points on the first cut, the second pair p_3-p_4 lists the points on the second cut, and the third pair lists the points on the third cut. For each pair always print the two points in alphabetical order. If there is a tie, print the solution which uses `A-B` over both `A-C` and `B-C`, and uses `A-C` over `B-C`.

Sample Input 1	Sample Output 1
50 50 25 10 45 20 20 40	110.66 A-C A-B B-C

Problem B
Diamonds Are for Evers

Top secret agent Claude L. Evers has a unique way to send encoded messages. The message is implanted in a square grid of letters as follows: starting at the middle letter at the left-most side of the grid he writes the message along the diagonal heading to the top of the grid. After he hits the top, he then continues the message on a diagonal heading right until he hits the right hand side. He continues this way, tracing out a diamond until he reaches the left side again. At this point he continues the message in the diamond nested inside the diamond he just filled in, and after this the next diamond inside, and so on. Any left over squares both inside and outside the original diamond are filled with random letters.

The grids below show the steps in encoding the message `GENERAL TSO NEEDS CHICKEN NOW`. The first grid shows the encoding after the first diamond has been filled in. The second, third and fourth grids show successively smaller diamonds being filled in. Finally, the last grid shows the final message after random letters have been placed in the empty squares around the grid.

			E			
		N		R		
	E				A	
G						L
	E				T	
		N		S		
			O			

(1)

			E			
		N	S	R		
	E	D		C	A	
G	E				H	L
	E	K		I	T	
		N	C	S		
			O			

(2)

			E			
		N	S	R		
	E	D	N	C	A	
G	E	E		N	H	L
	E	K	O	I	T	
		N	C	S		
			O			

(3)

			E			
		N	S	R		
	E	D	N	C	A	
G	E	E	W	N	H	L
	E	K	O	I	T	
		N	C	S		
			O			

(4)

T	H	E	E	S	E	A
R	E	N	S	R	N	T
A	E	D	N	C	A	C
G	E	E	W	N	H	L
T	E	K	O	I	T	U
A	L	N	C	S	L	Y
R	A	N	O	D	O	M

(5)

Note that if the original message had not filled in the entire set of diamonds then random letters would have been used to fill those empty slots as well (see Sample Input 2 for an example of this when encoding `GENERAL TSO HATES CHICKEN`). To send the message, Evers concatenates all the rows together, top-to-bottom. The above grid would be sent as

```
THEESEARENSRNTAEDNCACGEEWNHLTEKOITUALNCSLYRANODOM
```

Evers has made one fatal mistake, however: he just explained to us how he encodes messages! Now we can write a program that can decode any message he sends. And by "we" of course we mean you.

Input

Input consists of a single string of uppercase alphabetic letters. The length of each string will be a square of an odd integer and less than 1 000 in length.

Output

Output the decoded message. Note that this message might contain some random letters at the end if the original message did not fill the entire set of diamonds.

Sample Input 1

```
THEESEARENSRNTAEDNCACGEEWNHLTEKOITUALNCSLYRANODOM
```

Sample Output 1

```
GENERALTSONEEDSCHICKENNOW
```

Sample Input 2

```
WHAETHERENSRALLEENCAYGTELIHLKAKEITSARHCSEBIGMOACS
```

Sample Output 2

```
GENERALTSOHATESCHICKENIEL
```

Sample Input 3

```
JOMHNPARTAILUUHLASERINWGO
```

Sample Output 3

```
IAMTHEWALRUSU
```

Problem C
Flight Turbulence

Fly-By-Night Airlines is a low budget carrier specializing in late night/early morning flights. Things are run a little looser on this airline as was apparent a few nights ago. The flight from Ypsilanti, MI to Walla Walla, WA was completely sold out and all of the passengers were eagerly waiting to board the plane. Sensing their eagerness and looking for a little fun, the flight crew decided to open the gates and let the passengers rush to any seat they wanted, regardless of their seat assignment. The passengers were a little leery of this until the flight crew sweetened the deal with the promise of free peanuts for everyone. After a mad dash through the gates and one minor trampling injury all of the passengers had found a seat and were eagerly awaiting their peanuts.

Well, almost all. Wouldn't you know it, one passenger wasn't too happy with the seat he ended up with and decided he wanted his pre-assigned seat, peanuts be damned. When he threatened to call the FAA the flight crew relented and let him have his originally assigned seat, displacing the passenger in that seat. Well now, of course, THAT passenger wanted HER original seat. This dislodged another passenger who also wanted his original seat and so on. Eventually one of the dislodged passengers ended up in the seat initially vacated and things quieted down. Needless to say, the price of peanuts skyrocketed on that flight.

Some of the flight crew wanted to try this again on some other flights, but they were concerned if a similar situation might arise. They would like you to write a simulation program that would determine how many people would have to move if one person suddenly insisted on sitting in his/her assigned seat.

Input

Input starts with a line containing two positive integers n m where $n \leq 200$ is the number of passengers and $m \leq n$ is the current seat of a passenger who wants to sit in his/her assigned seat. The next line contains n integers. The value of the i^{th} of these integers indicates the assigned seat for the passenger who is currently sitting in seat i. All assigned seat numbers are unique and in the range $1 \ldots n$.

Output

Output the number of passengers that must change seats.

Sample Input 1	Sample Output 1
4 4 2 3 4 1	4

Sample Input 2	Sample Output 2
4 4 2 1 4 3	2

This page is intentionally left blank.

Problem D
Hardware Sales

Bree Kim Orter owns three hardware stores in different parts of the state. She's interested in knowing what items are popular across the state, specifically those items that have sold 20 or more units in each of her stores. For each store she has a chronological list of purchases made during the past week. Each purchase record consists of two integers: the ID number of the item and the number of units purchased. An example of such a list is shown below:

```
178293 3 457839 10 322228 1 ...
```

This list indicates that the first purchase that week was three units of the item with ID #178293. The next purchase involved 10 units of item #457839, and so on. Given three of these lists (one from each store) Bree would like to determine the number of items which sold 20 or more units in each store during the week.

Input

Input starts with a line containing three integers n_1 n_2 n_3, where $1 \leq n_1, n_2, n_3 \leq 100$ are the number of purchases in each of the three lists. Following this is the purchase list from the first store which contains n_1 pairs of positive integers, where the first number in each pair is a 6-digit ID number (possibly with leading zeros) and the second is the number of units sold. These n_1 pairs may span multiple lines. After this, starting on a new line, is the purchase list of the second store and following this, again starting on a new line, is the purchase list of the third store. All purchase amounts are ≤ 100. An ID number may show up multiple times on a list.

Output

Output the number of items that have sold at least 20 units in each of three stores. After this list the ID numbers of the items. Print the ID numbers in the order that they first appear in the input file.

Sample Input 1	Sample Output 1
5 2 4 123682 20 239481 23 001238 5 738299 4 001238 17 001238 31 123682 25 239481 25 123682 22 909090 18 001238 27	2 123682 001238

This page is intentionally left blank.

Problem E
Make America Grade Again

The University of America is having an integrity issue. It turns out that grades in many of the science classes are not being based on any graded work but on such things as attendance, grooming habits and any snacks the students bring with them. The administration has decided to crack down on this. First, they confiscated all of the snacks. Then they insisted that all grades in science classes must be on graded work in four categories: labs, homeworks, projects and exams. They don't care what percentage of the final grade is assigned to each of these as long as the total percentage assigned is 100%.

The professors at UA have grudgingly agreed to start grading their assignments, but they balk at the task of having to calculate final grades as well. They are asking you to write a program to calculate a student's final grade. They will provide you with the following information: first, the percentage of each of the four categories; second, a list of (grudgingly) graded assignments each of which fits into one of the categories. Using this, they want you to calculate a final integer percentage grade.

For example, suppose the assigned percentages are labs - 20, homeworks - 20, projects - 25 and exams - 35 and the list of graded assignments is the following:

Lab 1: $15/20$	Hw 3: $61/65$
Hw 1: $65/70$	Exam 2: $79/100$
Hw 2: $27/35$	Lab 2: $17/20$
Exam 1: $88/100$	Hw 4: $51/60$
Proj 1: $50/50$	Exam 3: $141/150$

To calculate the final grade you first total up the fraction of points achieved in each category. For labs it's $32/40$ (there were two assignments each worth 20 and the student got a 15 on the first and a 17 on the second for a total of 32 points); for homeworks it's $204/230$; for projects it's $50/50$ and for exams it's $308/350$. You then multiply each of these fractions by the percentage assigned to each category: $20(32/40) + 20(204/230) + 25(50/50) + 35(308/350) = 89.539$. Finally you truncate this to get the final grade of 89. Then you sit back and eat all the snacks you no longer need to bring to class.

Input

Input starts with five positive integers $l\ h\ p\ e\ n$, where the first four values are the percentages for labs, homeworks, projects and exams, respectively (always totaling to 100), and n ($4 \leq n \leq 200$) is the number of graded assignments. Following this are n lines each of the form $cat\ i$`:` r/s, where cat is either `Lab`, `Hw`, `Proj` or `Exam`, i is a positive integer, r is the number of points achieved for an assignment and s is the total possible points for the assignment. In each of these lines s is positive and $0 \leq r \leq s \leq 1\,000$. There will always be at least one assignment in each of the four categories.

Output

Output the final truncated percentage grade.

Sample Input 1

```
20 20 25 35 10
Lab 1: 15/20
Hw 1: 65/70
Hw 2: 27/35
Exam 1: 88/100
Proj 1: 50/50
Hw 3: 61/65
Exam 2: 79/100
Lab 2: 17/20
Hw 4: 51/60
Exam 3: 141/150
```

Sample Output 1

```
89
```

Sample Input 2

```
25 25 25 25 4
Exam 1: 100/100
Lab 2: 30/30
Hw 3: 50/50
Proj 4: 60/60
```

Sample Output 2

```
100
```

Problem F
This Problem's a Slam Dunk

It's getting close to March Madness time and the big game this week is the classic rivalry between State U and U State. The coach of State U has been working night and day on the best strategy to beat his arch rival. On defense he has decided to have his tallest player guard the opponent's tallest player, his second tallest player guard the opponent's second tallest player, and so on. He's a bit worried about the size differences between the two teams though, and needs to know for this defensive strategy how many of his players are taller than the person they're guarding. He's asked you to take time from writing term papers for his players to write a program to calculate this value for him.

Input

Input consists of two lines. The first line contains five positive integers specifying the heights of the five players on State U's team (in no particular order). The second line contains the similar information for the five players on U State's team (again in no particular order).

Output

Output the number of State U players taller than the person they are guarding assuming that the tallest State U player guards the tallest U State player, the second tallest State U player guards the second tallest U State player, and so on.

Sample Input 1	Sample Output 1
67 64 61 70 58 62 68 65 71 59	0

Sample Input 2	Sample Output 2
58 64 61 67 70 57 71 62 68 65	1

This page is intentionally left blank.