icpc | north america contests

The 2023 ICPC Southern California Regional Contest

Official Problem Set

JET BRAINS

icpc global sponsor
programming tools

upsilon pi epsilon
honor society

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 1
Fair Fare

The streets and avenues of Codeburg form a perfect Manhattan grid. The streets run east to west, perfectly parallel to each other. Similarly, the avenues run north to south. The distance between adjacent streets and avenues is the same.

The favorite mode of transportation in Codeburg is the taxi. It costs exactly $1 to go from an intersection to an adjacent intersection.

One day, Jay, your ICPC teammate, hailed a taxi at an intersection to go to Codeburg University, where the weekly ICPC practice sessions take place. To keep his mind sharp before the practice began, Jay looked outside the taxi window, memorizing every intersection he passed by.

After some time, Jay arrived at the school and paid the fare. However, he kept thinking that he might have overpaid for the trip to the school because the driver did not take the shortest possible route.

Jay sat down on his chair. Instead of implementing his favorite algorithm as a warm-up, he decided to write a program to compute how much he overpaid for the trip. You arrive right when Jay was about to start. Help him find out whether his taxi fare was fair.

The input is a series of 1 to 100 lines, terminated by end-of-file. Each line is a test case, with a single string consisting of uppercase letters 'L', 'R', and 'D', denoting Jay's trip to the school. The string will contain at least 1 and at most 1,000 letters.

The letter 'D' means that the taxi drove straight until the next intersection. The letter 'L' means that the taxi made a left turn at an intersection. Similarly, the letter 'R' means that the taxi made a right turn. Note that the letters 'L' and 'R' only indicate that the taxi made a turn, without driving to the next intersection.

For each test case, print on one line whether Jay overpaid for the taxi trip. If he overpaid by $x, then print "`overpaid by` x", without the quotation marks. If he did not overpay, then print "`fair fare`". Do not print leading or trailing whitespace on an output line.

Sample Input

```
DD
DRDLDLD
DRRRRD
```

Output for the Sample Input

```
fair fare
overpaid by 2
fair fare
```

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 2
Swamptalk

Many decades ago the students of Swamp County College built a small computer. It was pretty primitive and slow and had only a few instructions: add, subtract, halt, show, and some conditional branches.

They made a small computer language to test it: *Swamptalk*.

Your job is to implement the Swamptalk language. See the attached manual page.

The input consists of a Swamptalk program. A Swamptalk program itself has no input.

A program consists of a series of at most 100 lines, each of which is at most 80 characters long, terminated by end of file.

The judged data are guaranteed to be valid and not cause any integer underflow or overflow if arithmetic is done with at least 16 bits.

Sample Input

```
* mult
  set foo 10
  set bar 3
  set ans 0

moreMul: set ans ans + foo
    set bar bar-1
    gotoifp moreMul bar
show ans
* div/mod
  set foo 17
  set bar 3
  set ans 0

moreDiv: set foo foo - bar
    set ans ans+1
    gotoifp moreDiv foo -   bar
    show ans
    show foo

    gotoifp skip foo
    show 999
    halt
skip: set new new+foo-10
    show new
    set new new+foo-10
    show new
    halt
```

Problem 2
Swamptalk (continued)

Output for the Sample Input

```
30
5
2
-8
-16
```

SWAMPTALK

A Swamptalk program consists of instructions and comments.

A comment is a line consisting only of whitespace or a line where the first non-whitespace character is a '*'.

Instructions, one instruction per line, are made up of identifiers, keywords, literals, and operators. Elements on an instruction line are separated by whitespace and/or operators.

An identifier consists of a string of alphanumeric characters starting with an alphabetic character and is at most 16 characters long. Case is significant.

A keyword is one of: 'set', 'show', 'gotoifz', 'gotoifp', 'gotoifm', or 'halt'.

A literal is an integer constant: 0 <= literal <= 15000.

Operators are '+' and '-'.

An instruction consists of an optional label definition, a keyword, and various arguments depending on the keyword.

A label is an identifier and a label definition is a label immediately followed by a colon.

A variable is also an identifier. Labels and variables are in different address spaces and can be the same, although that is poor practice. Variables and labels can not be keywords. All variables are initialized to 0.

Expressions are a series of variables or literals separated by operators.

Execution begins with the first instruction.

Instructions:

set variable expression
  set the variable to the value of the expression

show expression
  print a line with the value of the expression
  print the integer with no unnecessary leading '0's, a '-' if necessary with no space between it and the integer. Do not print '+' signs. No leading or trailing spaces.

halt
  terminate execution

gotoifz label expression
  if the value of the expression is 0 continue execution at the statement with that label definition. Otherwise continue execution with the next statement.

```
gotoifp label expression
  if the value of the expression is >0 continue execution at the
  statement with that label definition. Otherwise continue execution
  with the next statement.

gotoifm label expression
  if the value of the expression is <0 continue execution at the
  statement with that label definition. Otherwise continue execution
  with the next statement.
```

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 3
Wormhole

In a distant future, when humanity has reached for the stars, a spaceship NebulaRunner is running away from space pirates.

The galactic map is a handy tool to navigate the vast space; it renders the space as a 2-dimensional grid on the x-y plane, divided into square-shaped sectors. NebulaRunner is currently at sector $(0, 0)$ (shown at the bottom left of the galactic map), and set its destination to a cosmic gateway at sector (n, m) (shown at the top right corner of the map).

NebulaRunner at sector (x, y) can move to any of its neighboring sectors $(x \pm 1, y)$ or $(x, y \pm 1)$. It takes 1 Space Time Unit (STU) for NebulaRunner to do so. The galactic map also showed k wormholes, each of which connects two different sectors. NebulaRunner can travel through a wormhole in 1 STU as if it travels between two adjacent sectors. Note that NebulaRunner can choose not to take a wormhole even if it is in the same sector as the spaceship.

The space pirates will have a harder time tracking NebulaRunner if there are many paths leading to the cosmic gateway. To estimate their chances of getting away safely, the captain of the spaceship wants to compute the number of different paths to the cosmic gateway that would also minimize the total travel time.

The input contains one or more lines. The first line contains n, m, and k ($1 \leq n, m \leq 25$, $0 \leq k \leq 10$), separated by whitespace. Each of the next k lines contains four space-separated integers x_s, y_s, x_t, and y_t ($0 \leq x_s, x_t \leq n$, $0 \leq y_s, y_t \leq m$), which describes a wormhole that connects sectors (x_s, y_s) and (x_t, y_t). Note that there can be multiple wormholes in the same sector. Wormholes always connect two distinct non-adjacent sectors. No two wormholes connect the same pair of sectors.

Print in a single line the number of distinct paths to sector (n, m) that minimizes the total travel time.

Sample Input 1

```
1 1 0
```

Output for Sample Input 1

```
2
```

Sample Input 2

```
2 2 0
```

Output for Sample Input 2

```
6
```

Problem 3
Wormhole (continued)

Sample Input 3

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

Output for Sample Input 3

```
2
```

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 4
Tontine Confusion

The State of Confusion has adopted a different kind of employee retirement investment plan. The State has implemented a *tontine*.

A tontine is a plan where a group of investors of similar age pool their funds for investment. The annual investment income is paid to the investors in proportion to their original investment. However, there is a catch...

When an investor dies, the income that accrues from that investor's shares is divided proportionally among the shares of the surviving members of the tontine. As time goes on, the surviving members get larger and larger annual payments!

In theory, in a tontine the last survivor gets everything left—"winner takes all". This could lead to both huge payments and "moral hazard" when only a few survivors remain. The State plan addresses these concerns by no longer decreasing the number of survivor's shares when the number of survivors drops to a given limit. The invested principal is never returned to the participants—when the last participant dies, the principal is kept by the State.

The State wants your team to write a program that, given a list of participants, their ages, their investment shares, and their longevity, will determine the per-share annual payments to the surviving participants at the end of each year. Each participant holds an integer number of shares valued at $100 each.

The plan separates the participants by age classes of a specified number of years based on their birth years. Each class is managed as a separate tontine. Age classes are based on participant ages at the end of the calendar year when the tontine first pays out. Age class number 1 starts at age 50 and the age classes are numbered in increasing order by age. For example, if the class interval is 10 years, class 1 is ages 50 through 59, class 2 is ages 60 through 69, and so on.

Input to your program consists of the following values, separated from each other by spaces and/or newlines:

- Four values P, M, B, and S. P is the total number of participants, between 1 and 2000 inclusive. M is the number of participants in a given age class at which the number of survivor's shares is no longer decreased due to participant deaths, between 1 and 25 inclusive. B is the number of birth years that make up a tontine age class, between 1 and 15 inclusive. S is the first payout year, between 2023 and 2060 inclusive.
- P value pairs, representing participants with ID numbers 1 through P in order. Each pair contains a participant birth year between $(S - 85)$ and $(S - 50)$ inclusive, followed by the integer number of $100 shares owned by the participant (between 1 and 20,000 inclusive).
- A list of values that represent each year of payouts to the tontine classes. The first value R is the annual return of the tontine classes, given as a percentage between 1 and 20 inclusive, with up to two digits after an optional decimal point. The second value D is the number of participants who died during the year, between zero and the number of remaining participants. D integers follow, each of which is an ID number of a deceased participant in the order that they died. The last list ends with the end-of-file, when there are no more living participants.

Problem 4
Tontine Confusion (continued)

For each year that there is at least one payout, your program is to print a line for each class with a payout containing the year, the ID number of the class, and the per-share payout to the survivors of that class in dollars and cents (without currency signs or thousands separators). The per-share payment amount is to be truncated to the cent. Years are to be printed in ascending order, as are the ID numbers of active classes for a given year. Values are to be separated by single spaces. Do not print entries for classes that no longer have living participants.

Sample Input

```
12  2  15  2024
1960 1500  1970 800  1953 3500  1966 175
1949 4700  1960 3500  1959 10100  1973 325
1939 6850  1956 12000  1951 8200  1946 1750
5.75 0 10 1 5
3.8 2 11 12 6 0 6 1 3
7.5 3 6 9 10 2.75 1 2
11 2 1 7 1.5 0 8.35 1 4
6.5 0 9 0
12 1 8
```

Output for the Sample Input

```
2024 1 5.75
2024 2 5.75
2024 3 5.75
2025 1 10.00
2025 2 11.32
2025 3 10.00
2026 1 3.80
2026 2 5.97
2026 3 3.80
2027 1 6.00
2027 2 9.43
2027 3 6.00
2028 1 6.00
2028 2 10.92
2028 3 6.00
2029 1 16.87
2029 2 13.65
2030 1 8.66
2030 2 5.00
2031 1 138.60
2032 1 18.90
2033 1 105.21
2034 1 81.90
2035 1 113.40
```

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 5
Rack Cables

GigantoCorp is expanding a large data center. The company is purchasing many servers, storage arrays, and network switches and routers to handle its growing requirements.

Most servers, storage arrays, and network switches today are installed in data center racks. The dimensions of the racks and the equipment installed in them are standard sizes; the standards are based on those from analog telephone equipment a century ago.

Equipment installed in a standard rack is 19 inches wide and of varying heights. The height of a piece of equipment is measured in integer *rack units*, where a rack unit ("U" for short) is 1.75 inches. A standard rack has an interior height of 42U, although it is possible to buy half-racks 21U tall or extra-height racks as tall as 55U. Each rack unit is numbered on the rack, starting with one at the bottom. These standards are used globally despite being based on U. S. customary units.

Cables need to be run between equipment installed in racks. These cables could be network cables between server ports and switches or patch panels, data cables between shelves of disk or solid-state drives and storage controllers, or other similar requirements. It is important to keep the cabling organized in order to make maintenance and future expansion of equipment in a rack easier. Cables come in standard metric lengths of 0.5, 1, 2, 3, and 5 meters. Short 0.5m cables are run directly from one port to another, but longer cables are routed along either side of the rack, depending on which routing is shorter. Figure 1 shows equipment mounted in the lower part of a rack, along with cable runs of varying lengths. There are 2.54 centimeters in an inch.

GigantoCorp is installing the equipment they are purchasing in racks. The staff know which equipment ports need to be connected to other ports within a rack, and they know at which rack unit each port will be at and how far from the left of the rack the port is. GigantoCorp wants your team to write a program that will tell them which cables to order.

Treat the side of the rack where the cables are run as a flat surface, and treat the ports as positioned vertically in the center of the rack unit where the port is located. Cable slack is provided to allow for minor variations in equipment as described below.

Input to your program is a series of one to sixty port connections, one per line, ending with end-of-file. Each connection is specified as two ports. Each port is a pair of values: the first value R is the rack unit number in the rack where the port is located ($1 \leq R \leq 55$), and the second value i is the number of inches from the left where the port is located (to the nearest tenth of an inch, $1.0 \leq i \leq 18.0$). Values are separated from each other by single spaces. No port connection is between two ports at the same rack unit. The cables shown in Figure 1 correspond to the sample input.

For each port connection, your program is to print a line giving the length in meters of the shortest standard cable that can connect the ports. Print the length as "`0.5`", "`1`", "`2`", "`3`", or "`5`". Each 0.5m cable must have a minimum of four inches of slack—that is, it must be at least four inches longer than the distance between the ports. For longer cables that are run on the side of the rack, an additional four inches needs to be allowed for the cable run, for a total slack of eight inches.

Problem 5
Rack Cables (continued)

Sample Input

```
2 5.5 12 1.4
1 5.5 12 17.6
10 3.7 8 4.4
8 6.5 6 3.7
```

Output for the Sample Input

```
1
2
0.5
0.5
```

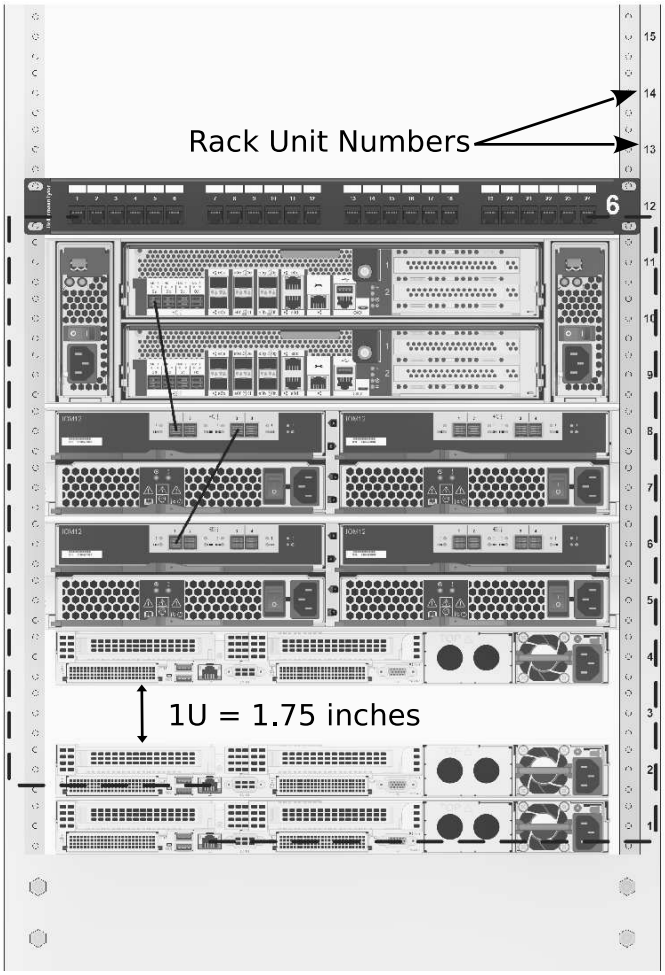

Figure 1. Rack diagram for the Sample Input. Short point-to-point cables are solid lines, cables along the side of the rack are dotted lines.

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 6
Asteroid

A giant triangular-shaped asteroid was discovered traveling at an alarming speed, right toward Earth!

As a researcher of the Space Force, you are given a mission of critical importance: compute how long it will take for the asteroid to touch the surface of Earth. That is the only time humanity has left to prevent the disaster. There is no time for complex modeling, so you have come up with a simple model that captures the essence of the situation.

The asteroid is modeled as a triangle on the 2-dimensional plane, with vertices (x_1, y_1), (x_2, y_2), and (x_3, y_3). It rotates around its center of mass, which is at $(0, 0)$. The center of mass lies strictly inside the triangle.

The asteroid is traveling directly along the x-axis, at a speed of v. In other words, the center of mass moves to the right by v units per unit time.

At the same time, the asteroid rotates around its center of mass counter-clockwise, completing a full circle every T units of time.

Earth is large enough that its surface is modeled as a straight line at $x = W$. When any part of the triangle-shaped asteroid touches the line, it is considered to have touched the surface of Earth.

Given this information, compute how many units of time it will take for the asteroid to reach the surface of Earth.

The input contains two lines. The first line contains 6 space-separated integers x_1, y_1, x_2, y_2, x_3, and y_3. All coordinates have an absolute value of at most 10,000. The second line contains 3 space-separated integers v, T, and W ($0 < v, T <$ 10,000 and $0 < W <$ 200,000). You are guaranteed that the asteroid has not touched the surface of Earth already. In other words, $x_1, x_2, x_3 < W$.

Print on one line the units of time it will take for the asteroid to reach the surface of Earth. Values within 10^{-5} (relative or absolute) of the judges' reference values are considered correct.

Sample Input 1

```
5 3 -3 3 -3 -5
15 4 20
```

Output for Sample Input 1

```
1.00000
```

Problem 6
Asteroid (continued)

Sample Input 2

```
10 0 0 5 -7 -7
30 20 50
```

Output for Sample Input 2

```
1.3634477991
```

Sample Input 3

```
10 0 0 5 -7 -7
30 1 50
```

Output for Sample Input 3

```
1.3432343213
```

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 7
Balance of Power

There are n legislators in the State of Confusion, each representing one of the three major parties: *Future One*, *Two-gether*, and *Triple Harmony*.

The founders of the State envisioned a healthy society where the three parties maintain the balance of power and no party gets a dictatorial position by having too many legislators. Formally, we say that the balance of power is achieved when no one party has strictly more legislators than the other two parties combined.

As the major election cycle for the State of Confusion is wrapping up, every journalist from the media is rushing to be the first to report the outcome. Fortunately, you, a journalist from the Profound Confusion Network, have some programming experience. Instead of counting and doing the math on paper, you decide to write a program to help you write an article. You will simply provide the party affiliation of the election winners, and the program will report whether the balance of power is achieved or not.

The first line of input to your program contains a single integer n ($3 \leq n \leq 80$), which represents the number of legislators in the State of Confusion. The next line of input contains n space-separated integers, representing the party affiliation of each elected person. Number 1 means a legislator of Future One. Similarly, the numbers 2 and 3 mean a legislator of Two-gether and Triple Harmony, respectively.

The output is a string on a single line, representing the headline of the article you will write. If the balance of power is achieved, then print only "`Power Balanced`" without quotation marks. Otherwise, print only "*[Party]* `Dominates`" without quotation marks, where "*[Party]*" is replaced with the name of the winning party. Note that the output is case-sensitive, and must match the format exactly without leading or trailing whitespace.

Sample Input 1

```
5
1 2 1 3 1
```

Output for Sample Input 1

```
Future One Dominates
```

Sample Input 2

```
10
1 2 1 3 1 2 3 3 1 2
```

Output for Sample Input 2

```
Power Balanced
```

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 8
Parabolic Triples

A Pythagorean triple is a set of three integers (x, y, z), each greater than zero, such that $z^2 = x^2 + y^2$.

Note that if (x, y, z) is a Pythagorean triple, then so is (ax, ay, az) for any integer a greater than zero since $a^2z^2 = a^2x^2 + a^2y^2$. If x and y are relatively prime (that is, they have no common factors greater than one), then we call that triple a *Primitive Pythagorean Triple (PPT)*.

If we plot the points (x, y) of PPTs up to some maximum the resulting plot has some interesting patterns. See Figure 1. The curvy lines can be shown to be parabolas, each with a vertex on the x or y axis and all with the focus at $(0, 0)$. (Why they are parabolas could be the subject of a math contest problem.)

One such parabola with the vertex at $(0, 2116)$ is plotted in gray. The PPTs on it are marked with 'X's. Note that there are other nearby PPTs, but they are not on the parabola.

Your team is to write a program that counts the number of PPTs on parabolas with vertices on the positive y axis and the focus at $(0, 0)$. Each parabola will be specified by the vertex point on the y axis: $(0, k)$.

The equation of such a parabola is:

$$x^2 = -4k(y - k)$$

Input to your program is a series of one to fifty lines, each of which is a value of k, $0 < k \leq 10{,}000$, terminated by end of file.

For each value of k, print a line containing k followed by the number of PPTs on the parabola with the vertex at $(0, k)$, separated by a space.

Sample Input

```
2116
16
3249
9409
2001
2500
9604
3136
8
```

Problem 8
Parabolic Triples (continued)

Output for the Sample Input

```
2116 22
16 2
3249 18
9409 48
2001 0
2500 20
9604 42
3136 24
8 0
```

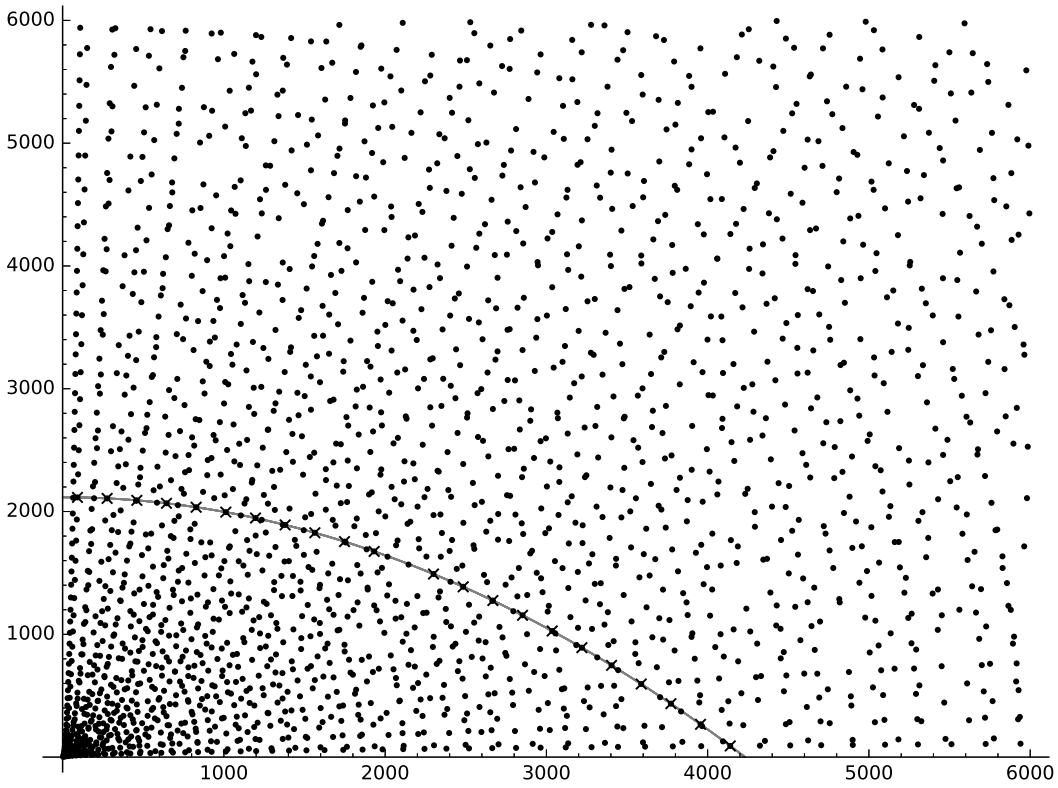

Figure 1. Plot showing parabola with vertex at $(0, 2116)$, corresponding to the first line of the sample input.

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 9
Fielding Soccer Players

An American Youth Soccer Organization (AYSO) soccer game is divided into four quarters rather than two halves. In each quarter, F players in the team roster are fielded. To satisfy one of AYSO's principles, "Everyone Plays," each player on a team's roster must play at least two quarters per game. To assist the coach in fielding the best overall combination of players, you have been tasked to compute the best team of fielded players over all quarters based upon each individual player's skill level, as well as the joint performance of combinations of players.

Given a list of players, their individual and joint skill levels, and the AYSO rule that all team members must play at least half the game (two quarters), field the strongest team across all four quarters. The strongest fielded team is the sum of the individual and joint skills of each quarter's fielded players. S_1 through S_4 are the team's skill levels for each quarter. The strongest fielding of players is acheived when $S_1 + S_2 + S_3 + S_4$ is maximized, according to F and the Everyone Plays rule.

Input to your program is a series of lines.

- The first line contains a single integer, P, the number players on the team roster. $5 \leq P \leq 14$.
- The second line contains a single integer, F, the number of players on the field at any one time. $P/2 < F \leq \min(P, 11)$.
- The next P lines represent the roster of players in ascending order by jersey number, one player per line. Each player is specified with a the jersey number, j, a comma, and the name. $2 \leq j \leq 99$.
- The remaining 0 to 50 lines until end-of-file represent individual and joint skill level adjustments. Skill level adjustments are a series of whitespace delimited integers. The first integer is the skill level, s. $-50 \leq s \leq 50$. The second integer k is the number of jersey numbers to follow. $1 \leq k \leq F$. The remaining k integers on the line are jersey numbers.

There is no particular ordering of the skill level adjustment lines. When evaluating the combined team strength per quarter, S_q, apply all combinations that occur for the fielded team members. Combinations that include players *not* fielded are *not* applied.

Skill levels show what each individual player or player combination contribute to the team. Some players are individually better (positive adjustment), individually worse (negative adjustment), and certain combinations can be positive or negative. For instance, two players might make a great pair of strikers who cooperate and strengthen the team, whereas a trio of divas might compete against each other, bringing the team performance down. In the sample input, Neymar (jersey 9) and Ada (jersey 13) are individually skilled (additional 5 and 4, repectively), but they do not play well together; their net effect when fielded in the same quarter is $5 + 4 - 3$. Players without individual skill ratings contribute 0. When a skill adjustment is present for more than one player, the player jersey numbers will occur in ascending order. No combinations of players will ever repeat in the skill section.

Print a single integer that maximizes $S_1 + S_2 + S_3 + S_4$, subject of course to the rule that each player must play at least two quarters.

Problem 9
Fielding Soccer Players (continued)

Sample Input

```
11
9
2,Lionel Smith
3,Lucy Davenport
5,Cristiano Ramirez
6,Javi Ruiz
7,Sam Meade
8,Pernille Abramovich
9,Neymar Jones
10,Wendie Goetter
12,Vincent van Vliet
13,Ada Lovelace
14,Chloe Pele
2 1 14
4 1 13
5 1 9
1 1 2
2 1 7
3 2 13 14
-1 1 5
-10 1 3
-3 2 9 13
7 3 2 6 10
```

Output for the Sample Input

```
62
```

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 10
Swamp County Title Company

Forty years ago, the Pair-O-Dice Real Estate company subdivided a Swamp County section of vacant land (a square one mile on a side, aligned on compass points, for a total of 640 acres) into parcels. Each parcel was a subdivision of a section (called an "aliquot part") taken by halving or quartering the section, or halving or quartering such a fraction. Fractions are either the letter of a cardinal direction (north (N), south (S), east (E), or west (W)) followed by "1/2", or letters for an ordinal direction (northeast (NE), southeast (SE), southwest (SW), or northwest (NW)) followed by "1/4". Examples include *"N1/2"*, *"SE1/4"*, *"NE1/4 OF NE1/4"*, *"E1/2 OF SW1/4 OF NE1/4"*, and so forth.

It was discovered at the time that Pair-O-Dice did not keep good records and sold some parcels to unwitting buyers that overlapped other parcels. The ICPC teams of the day wrote programs to detect the overlapping parcels. However, those programs are long gone, as are the computers they ran on. What is more, it seems that parcels in other square-mile sections within the same six-mile by six-mile square township may also suffer from similar problems! Some of those other parcels may be entire sections. Figure 1 shows how the U. S. Public Land Survey System divides a square township six miles on a side into square sections one mile on a side, and Figure 2 shows one possible fractional division of a section.

The Swamp County Title Company needs to sort all this out so they can determine which titles are valid. The company wants your team to write a program that will do this for them.

Your program is to read a series of lines that describe parcels in the order they were sold, and determine for each parcel description whether it overlaps a previously sold valid parcel or not. Each description is zero to five fractions (with multiple fractions separated by "`OF`") followed by the word "`SECTION`" and an integer section number in the range 1 to 36 inclusive. No parcel crosses a section boundary. Words and fractions are in upper-case and are separated from each other by one or more spaces. No input line exceeds 80 characters. There will be between 1 and 500 parcel descriptions in the input. Input ends with the end-of-file.

Your program is to check each description in turn to determine if it overlaps any parcel already determined to be valid. (The first parcel is always valid.) If the parcel description does not overlap any previous valid parcel, add it to the valid parcel list by assigning a sequential parcel number starting at 1 and print a line giving the parcel number and the size of the parcel in acres. If the parcel size is not an integer number of acres, print the number of whole acres (if greater than zero) followed by the fractional portion in the form a/b in lowest terms. Follow this by the word "`ACRE`" for sizes of one acre or less, or "`ACRES`" for larger parcels. Use single spaces to separate the output fields.

If the parcel description overlaps a valid parcel, print a line containing only the string "`OVERLAPS`" followed by a single space and the lowest valid parcel number it overlaps.

No leading or trailing whitespace is to appear on an output line.

Sample Input

```
NE1/4 SECTION 1
W1/2 OF SE1/4  SECTION 1
E1/2  SECTION 1
N1/2  OF SW1/4 OF SW1/4  SECTION 1
N1/2  OF SW1/4 OF NW1/4  SECTION 1
N1/2  SECTION 2
NE1/4 OF N1/2  SECTION 2
NW1/4 OF SW1/4 OF NE1/4 OF SE1/4 SECTION 2
```

Problem 10
Swamp County Title Company (continued)

Output for the Sample Input

```
1 160 ACRES
2 80 ACRES
OVERLAPS 1
3 20 ACRES
4 20 ACRES
5 320 ACRES
OVERLAPS 5
6 2 1/2 ACRES
```

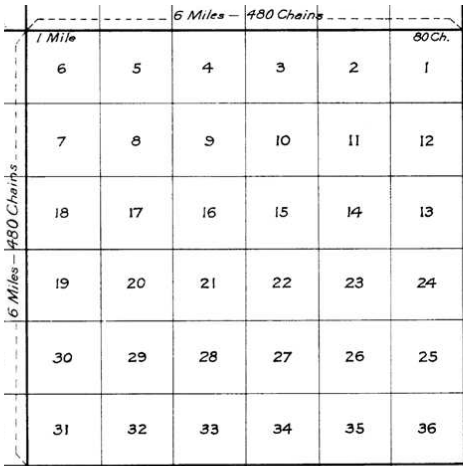

Figure 1. Township divided into square sections.

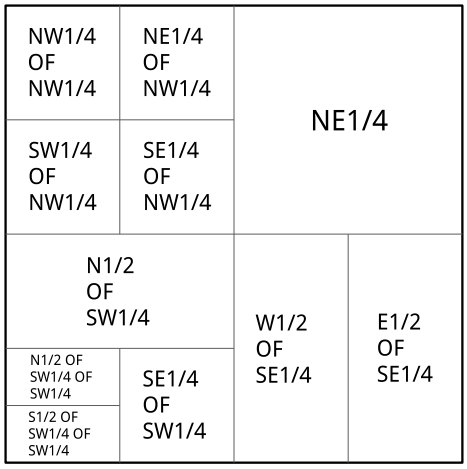

Figure 2. Sample division of a section into fractional aliquot parts.

2023/2024 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 11
Exciting Tournament

A total of n players enter a single-elimination tournament, where $n = 2^k$ for some integer $k \geq 1$. The tournament proceeds in k rounds. At each round, all remaining players are paired up at random and play a match. The winner of each match advances to the next round, while the loser is eliminated from the tournament.

Each player is assigned an ID 1 (strongest) to n (weakest). We assume that when two players play in a match, the player with the smaller ID will always win.

Between every pair of players, there is an *excitement score* associated with it. The excitement score of the tournament is equal to the sum of the excitement score in every match. Assuming that the tournament matchups are chosen uniformly at random for each round, find the expected total excitement score of the tournament.

The first line of the input contains a single integer n ($2 \leq n \leq 64$), denoting the number of players in the tournament. Each of the next n lines contains the table of excitement scores. The jth integer on the $(i+1)$st line of input is the excitement score e_{ij} between players i and j ($-100 \leq e_{ij} \leq 100$). You are guaranteed that $e_{ij} = e_{ji}$ for all i and j, and that $e_{ii} = 0$ for all i.

Print on one line the expected total excitement score of the tournament, when the tournament matchups are decided randomly. Values within 10^{-5} (relative or absolute) of the judges' reference values are considered correct.

Sample Input 1

```
4
0 15 4 -2
15 0 10 1
4 10 0 8
-2 1 8 0
```

Output for Sample Input 1

```
23.3333333333
```

Sample Input 2

```
8
0   10  8  -7 16  8  5  -14
10  0  -4  0  16  -1 7  2
8   -4  0  12 15  13 9  0
-7  0   12 0  -1  14 7  -6
16  16  15 -1 0   -3 2  -11
8   -1  13 14 -3  0  17 20
5   7   9  7  2   17 0  16
-14 2   0  -6 -11 20 16 0
```

Problem 11
Exciting Tournament (continued)

Output for Sample Input 2

```
41.28571428
```