November 2015

**Computer Science Competition**

Hands-On Programming Set

**I. General Notes**

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

**II. Point Values and Names of Problems**

|  |  |
| --- | --- |
| **Number** | **Name** |
| Problem 1 | Wake Up! |
| Problem 2 | Customer Scheduling |
| Problem 3 | Joe’s Agenda |
| Problem 4 | Shopping List |
| Problem 5 | Voltage and Power |
| Problem 6 | Flipping Lights |
| Problem 7 | Gates |
| Problem 8 | Electricity Flow |
| Problem 9 | Fixing Appliances |
| Problem 10 | Pay in Cash |
| Problem 11 | The Way Back Home |
| Problem 12 | Key Probability |

**1. Wake Up!**

# Program Name: Alarm.java Input File: no input file

Joe is just an average guy with an average job and an average wake up time. Draw Joe’s alarm clock at 7:00 A.M.

**Input**

This problem has no input.

**Output**

Reproduce the ascii art picture of Joe’s alarm clock on the screen. the numbers along the top and left side of the example output are for convinience only and should not be included in the output

**Example Output to Screen**

01234567890123456789012345678901234567890123

1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /|

3| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |

4| | | | |

5| | \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* | | |

6| | \*\*\*\*\*\*\*\* \*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* | | |

7| | \*\* \*\* \*\* \*\* \*\* \*\* | | |

8| | \*\* \*\* \*\* \*\* \*\* | | |

9| | \*\* \*\* \*\* \*\* \*\* | | |

0| | \*\* \*\* \*\* \*\* \*\* \*\* | | |

1| | \*\* \*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* | | |

2| | \*\* \*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\* | | |

3| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| | |

4|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|/

**2. Customer Scheduling**

# Program Name: Schedule.java Input File: schedule.dat

Joe is an electrician. Every day, Joe has a list of jobs that he has been offered. Being the great electrician that he is, Joe is often overbooked, and is forced to choose which jobs to take in order to maximize the amount of jobs he can do in a day. Joe has enlisted you to create a computer program to help him with this process.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with a single integer x denoting how many jobs Joe has been offered. The following X lines will contain a start and end time formatted HH:MM AM/PM. Joe will not accpet a job beginning before 03:00 AM or after 09:00 PM.

**Output**

Output the maximum number of jobs that Joe can accept that day without any times overlapping. If one event starts at the same time as another ends, the two events are not considered overlapping.

**Example Input File**

1

7

01:30 AM 02:00 AM

06:30 AM 07:00 PM

06:45 AM 08:30 AM

07:30 AM 09:00 AM

08:45 AM 09:15 AM

09:07 AM 01:00 PM

09:20 AM 01:00 PM

**Example Output to Screen**

3

**3. Joe’s Agenda**

# Program Name: Agenda.java Input File: agenda.dat

Joe’s schedule today is totally booked. On days like this, he likes to know exactly what time he’ll be finished. Assuming that Joe will either be working or driving nonstop througout his workday, use his schedule to predict when he will finally arrive home from his workday.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with an integer s followed by a time in the format HH:MM AM/PM, representing the speed of Joe’s car, and the time Joe begins the first item on his to do list. An unknown number of items will follow. The following lines will each contain one item of Joe’s to do list, formatted by the items name, followed by either a distance in miles or a time in minutes, separated from the name by a comma. The time it takes to complete that item will either be the time it takes for Joe’s car to travel the provided distance, or the time listed. each case is terminated by the “GO HOME” item.

**Output**

Output the time joe will arrive home to the nearest minute in the following format :

"Joe will arive home at HH:MM AM/PM".

**Example Input File**

2

10 07:00 AM

GO TO JOB 1, 20 MILES

DO JOB 1, 30 MINUTES

GO TO JOB 2, 40 MILES

DO JOB 2, 45 MINUTES

GO HOME, 10 MILES

50 05:00 PM

GO TO STORE, 25 MILES

SHOP, 15 MINUTES

GO HOME, 25 MILES

**Example Output to Screen**

Joe will arrive home at 03:15 PM

Joe will arrive home at 06:15 PM

**4. Shopping List**

# Program Name: List.java Input File: list.dat

Being the fantastic electrician that he is, Joe volunteers to provide his own supplies for whatever jobs he performs. With all of his job opportunities he often runs out of supplies though. Joe is making a trip to the store, and has enlisted you to write a program to calculate how much money he will spend.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with two integers n and m, n being how many items the store sells, m being how many items are on Joe’s list. The following n lines will each contain an item name, item quantity per package, and cost per package, all separated by commas. The next m lines each contain an item on Joe’s grocery list, followed by the quantity he needs, separated by a comma. It is also worth noting that Joe will sometimes have to buy extra, as the package size may not divide evenly into the quantity needed, and that the tax rate on all of Joe’s purchases is 6.25%

**Output**

Output the cost of Joe’s shopping trip in the format:

“Joe’s trip to the store costs him ” followed by the cost of his trip rounded to the nearest cent.

**Example Input File**

1

6 5

Light Bulbs, 6, 1.50

Screws, 50, 2.00

Screwdriver, 1, 5.00

Hammer, 1, 6.00

Nails, 50, 2.50

Wrenches, 16, 40.00

Light Bulbs, 15

Screwdriver, 1

Nails, 250

Wrenches, 17

Hammer, 1

**Example Output to Screen**

Joe’s trip to the store costs him $114.75

**5. Voltage and Power**

# Program Name: Voltage.java Input File: voltage.dat

Being the stellar electrician he is, Joe happens to have a firm grasp on the concepts of current, voltage, resistence, and power. He knows the following two equations by heart:

**V = IR P = IV**

V, I, R, and P denoting voltage, current, resistence, and power respectively. Joe finds doing the math in solving for these variables repetitive though, and has enlisted you to help write a program to do it for him

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will contain two lines. each line will give you the value of either V,I,R, or, P.

**Output**

find the values of V,I,R, and P, and print them in the format:

“V = X.XXX, I = X.XXX, R = X.XXX, P = X.XXX”, all rounded to three decimal places.

**Example Input File**

3

V = 5

I = 10

P = 5

V = 2.5

I = 5

R = 5

**Example Output to Screen**

V = 5.000, I = 10.000, R = 0.500, P = 50.000

V = 2.500, I = 2.000, R = 1.250, P = 5.000

V = 25.000, I = 5.000, R = 5.000, P = 125.000

**6. Flipping Lights**

# Program Name: Lights.java Input File: lights.dat

Joe is working a job where he has to test a lot of light switches. given an initial position of a row of light bulbs and instructions on how to flip the switches, help show Joe how the row should look after he’s done.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with a string representing the row of light bulbs, 1 being on and 0 off, and a single integer m representing the number of actions to be performed on the row of lights. There are 6 possible actions:

* FLIP A B – flips all of the lights to their inverse from A to B exclusive
* FLIP ALL – flips all of the lights to their inverse
* ON A B – turns on all lights from A to B exclusive
* ON ALL – turns on all lights
* OFF A B – turns off all lights from A to B exclusive
* OFF ALL – turns off all lights

**Output**

Output what the string of lights should should look like after all of the actions have been performed.

**Example Input File**

2

1010101010 4

FLIP ALL

ON 0 2

ON 8 10

OFF 4 6

000000 2

ON 0 3

FLIP ALL

**Example Output to Screen**

1101000111

000111

**7. Gates**

# Program Name: Gates.java Input File: gates.dat

Logic gates are important in the world of electricianeering. Joe has been studying up on his logical operators, drawing diagrams and truth tables and whatnot. write a program to generate a truth table for a given boolean statement so Joe can check his work

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with a single integer x denoting how many variables are in the following statement, followed by a boolean expression consisting of !,&,|,^, and letters A-G. A letter will not appear in the string unless the letter preceeding it has already occurred in the string as well. For example, there will be no test case B&D, as B occurs without an A, and there will be no test case B&A, as A did not preceed B. Follow order of operations. There will be no parenthesis.

**Output**

output the truth table for the given boolean expression. The first column of the truth table should represent A, the second B, the third C, and so on. The truth table must be in binary order. For example in the first test case, if you were to to replace the boolean values of A,B, and C with 1’s for true and 0’s for false, the boolean combination with the smallest binary representation would have to come first. Columns also must be properly alligned with width of 6.

**Example Input File**

2

3 A&B^C

2 A|B

**Example Output to Screen**

FALSE FALSE FALSE FALSE

FALSE FASLE TRUE TRUE

FALSE TRUE FALSE FALSE

FALSE TRUE TRUE TRUE

TRUE FALSE FALSE FALSE

TRUE FALSE TRUE TRUE

TRUE TRUE FALSE TRUE

TRUE TRUE TRUE FALSE

FALSE FALSE FALSE

FALSE TRUE TRUE

TRUE FALSE TRUE

TRUE TRUE TRUE

**8. Electricity Flow**

# Program Name: Flow.java Input File: flow.dat

Joe is working on wiring some complicated electrical contraptions together. In order to do this, he needs to know the maximum amount of electricity that can flow from the power source to its destination. He’s using many leftover wires that have different current carrying capacities.

3

2

3

4

2

3

3

2

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with two space separated integers x and y, x representing the number of junctions and y the number of wires. The next y lines will each contain three space separated numbers A,B, and C, A being the start junction of the edge, B being the end junction of the edge, and C being the maximum amount of current the wire can transfer.

**Output**

find the maximum amount of current that can pass through the wire network from the start to destination. Start will always be junction 0, and destination will always be junction x-1. it is legal to split ones flow between two paths. for example, node 0 can send 3 units of electricity to one child and three to the other. if you are only receiving 3 units however, you cannot pass any more than a total of 3 to all of the following nodes. Print out the maximum current in the following format:

“The maximum flow is X”

**8. Electricity Flow(Continued)**

# Program Name: Flow.java Input File: flow.dat

**Example Input File**

1

6 8

0 1 3

0 2 3

1 2 2

1 3 3

2 4 2

3 4 4

3 5 2

4 5 3

**Example Output to Screen**

The maximum flow is 5

**9. Fixing Appliances**

# Program Name: Fixing.java Input File: fixing.dat

Joe has been hired to fix a microwave, washer, dryer, and air conditioning, all in one job. He is in a rush however, and is wondering what would be the fastest way to finish all of these jobs as to get going. Write a program to help Joe solve his problem.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with a single integer x denoting the size of the floorplan of each house.

the next x lines will represent the floorplan of the house, J being Joe’s starting position, M being the microwave, W being the Washer, Dbing the Dryer, and A being the Air Conditioning, and a # being a wall that obstructs Joe’s movement. Joe can only move up, down, left, or right

**Output**

Output the shortest time in which Joe can fix all 4 appliances. it takes one second for Joe to take a step from one square in the floorplan to another, and it takes Joe 10 seconds to repair an appliance.

**Example Input File**

2

5

#####

#M.J#

#..W#

#D.A#

#####

9

#########

#J.....M#

####W####

#....#A.#

#.#####.#

#.#D....#

#.#####.#

#.......#

#########

**Example Output to Screen**

46 seconds

76 seconds

**10. Pay in Cash**

# Program Name: Cash.java Input File: cash.dat

After a tough job, Joe asks his customer for payment. Joe doesn’t like credit, debit, or check. Joe likes cold, hard, cash. The problem is, whenever Joe gives his customers their fee, customers will spend entirely too long sorting through their money attempting to pay in exact change. Help Joe by writing a program to see if this is possible.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with two space separated integers,x denoting the number of coins in the customers hand, and y denoting the amount of change the customer is still trying to make. the next line will consist of x integers, representing each coin in the customers hand.

**Output**

Depending on whether or not it is possible for the customer to make change of Y cents, output either “Y is possible” or “Y is not possible”

**Example Input File**

2

10 99

25 25 25 10 10 10 5 5 1 1

10 99

25 25 25 10 10 10 1 1 1 1

**Example Output to Screen**

99 is not possible

99 is possible

**11. The Way Back Home**

# Program Name: Home.java Input File: home.dat

Joe’s day is finally done,and he can now head back home to relax. The only problem is, he doesn’t exactly know how to get home from his last job. Joe doesn’t trust GPS, so it is your job to write a program to help him home. Luckily he remembers the order of turns he made. Help Joe find his way home exactly the way he came.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will start with a single integer x denoting how many turns Joe made. The next line will consist of x Strings of either “left” or “right”.

**Output**

Output which ways to turn to get Joe home.

**Example Input File**

2

10

left right left right left left left right right left

4

left left left right

**Example Output to Screen**

right left right left right right right left left right

right right right left

**12. Key Probability**

# Program Name: Keys.java Input File: keys.dat

After a long and tiring day of work, Joe arrives home. The sky is dark and all is quiet around him. The darkness and fatigue from his hard work has left Joe unable to make out much detail from anything. Joe pulls out his key ring, and thinks to himself “maybe if I just put a random key in my door, I’ll get lucky and it’ll be the right one”. Joe is very tired. All Joe wants to know is what his odds of picking the correct key on the first try are.

**Input**

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will consist of a single integer x denoting how many keys Joe has on his key ring.

**Output**

Output the likeliness of Joe picking the correct key on the first try, rounded to two decimal places followed by a percent sign.

**Example Input File**

3

1

2

8

**Example Output to Screen**

100.00%

50.00%

12.50%