**A+ Computer Science**

**January 2015 – Packet 1**

Computer Science Competition

Hands-On Programming Set

DO NOT OPEN THIS PACKET UNTIL INSTRUCTED TO DO SO

1. General Notes
2. Do the problems in any order you like. They do not have to be done in order from 1 to 12.
3. All problems have a value of 60 points. Incorrect submissions may be reworked and resubmitted, but will receive a deduction of 5 points for each incorrect submission. Deductions are only included in the team score for problems that are ultimately solved correctly.
4. 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.
5. Your program should not print extraneous output. Follow the form exactly as given in the problem.

|  |  |
| --- | --- |
| Number | Name |
| Problem 1 | Determined1 |
| Problem 2 | Microwave |
| Problem 3 | Splatter |
| Problem 4 | BlackJack |
| Problem 5 | Drought |
| Problem 6 | Cross |
| Problem 7 | Pascal |
| Problem 8 | GPA1 |
| Problem 9 | Buoyancy |
| Problem 10 | Between |
| Problem 11 | GPA2 |
| Problem 12 | Determined2 |

Good luck!

1. Determined1

Program Name: Determined1.java Input File: determined1.dat

You are determined to find the determinant of a 2x2 matrix. A matrix is a set of numbers in rows and columns. A 2x2 matrix determinant can be found by:

Given 2 rows and 2 columns of numbers, find the determinant.

Input: The first line (N) consists of the number of data sets. Each data set contains 2 rows and 2 columns of integers.

Output: Print out each determinant on one line.

Constraints:

1<=N<=10

**Example Input File**

3

1 2

3 3

10 10

10 10

-2 3

-4 -5

**Output to screen:**

-3

0

22

2. Microwave

Program Name: Microwave.java Input File: microwave.dat

I love popcorn—kettle corn, buttered popcorn, caramel corn, etc. However, each different type has different microwave cooking times, especially for the smaller single bags. Sometimes they have the directions in seconds, and other times in minutes, or both. When you type in the time on MY microwave’s keypad, sometimes it takes it in seconds, and sometimes in minutes. For example, if I type in “60”, then my microwave will cook for a minute. If I type in “90”, then it will cook 1 minute 30 seconds. If I type in “100”, it shows as 1 minute and counts down “01:00, 00:59, 00:58...” In other words, times with 1 or 2 digits represent seconds only. Times with 3 or 4 digits (my microwave shows 4 digits on the display) are shown as **MM:ss**, where **MM** is in minutes and **ss** in seconds. Write a program that will show each time as 1) the number of seconds and 2) minutes:seconds in the format **MM:ss**.

Input: The first line (N) consists of the number of data sets. Each data set contains 1 integer, T, the microwave input time.

Output: Print out the time in the form: “\_\_seconds = MM:ss minutes.”

Constraints:

1<=N<=10

1<=T<=9959

**Example Input File**

8

60

59

99

100

130

200

10

1000

**Output to screen:**

60 seconds = 01:00 minutes.

59 seconds = 00:59 minutes.

99 seconds = 01:39 minutes.

60 seconds = 01:00 minutes.

90 seconds = 01:30 minutes.

120 seconds = 02:00 minutes.

10 seconds = 00:10 minutes.

600 seconds = 10:00 minutes.

1. Splatter

Program Name: Splatter.java Input File: splatter.dat

When I last painted the rooms in my house, my youngest daughter was 3 years old. I thought it would be funny if she helped by “throwing” paint on the wall, like a Jackson Pollock painting. Let’s write a program simulating this! You will be given a wall size (matrix) and some paint splatters. When the paint hits the wall, it spreads out (2 up, down, left, right and 1 diagonally).

Here are examples of 2 splatters at (3,4) and (0,8):

(0,0)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | X | X | X | X |
|  |  |  |  | X |  |  | X | X | X |
|  |  |  | X | X | X |  |  | X |  |
|  |  | X | X | X | X | X |  |  |  |
|  |  |  | X | X | X |  |  |  |  |
|  |  |  |  | X |  |  |  |  |  |

(5,9)

A wall is considered covered if there is a splatter on every square foot.

Input: The first line (N) consists of the number of data sets. Each data set starts with the size of the wall (R and C). The next line contains the number of paint splatters, T. Then is followed by T coordinates of the splatters (X and Y).

Output: Print out “YES” or “NO” to indicate if the wall is covered or not.

Constraints:

1<=N<=10

1<=R<=20

1<=C<=20

1<=T<=R\*C

**(Continued on next page…)**

**(Problem 3 continued)**

**Example Input File**

2

6 10

2

3 4

0 8

4 4

4

1 1

3 3

3 1

1 3

**Output to screen:**

NO

YES

4. BlackJack

Program Name: BlackJack.java Input File: blackjack.dat

Blackjack, or 21, is a common card game. You and the dealer are trying to get a card value of 21. Face cards (J, Q, K) are worth 10 and number cards are worth their number value. An ace (A) could be either 1 or 11 depending on how big your hand is. For example, a hand of [2, 9, A] would have a total of 12, since 2+9+11=22 would be a bust. A hand of [K, A] is 21, blackjack!

For this program, you want to write a game that is the computer playing against a person. You write the code for the computer, acting like the dealer.

The dealer in this program can only make a semi-intelligent decision to hold or draw. It must hold on 17 or more and must hit (draw a card) on 16 or less. That would mean a hand of [2, 9, A] would have a total of 2+9+1 = 12, and would have to hit because it was 16 or less. A card of 9 would be the best to make 21, and a 10 or higher would bust. Here the ace must be a 1 because 11 would make a bust of 22.

Here is the algorithm for the dealer:

* Draw a card if the total is 16 or less.
* Hold if the total is 17 or more.
* An ace counts for 11 if the total is 10 or less, or counts for 1 if not. If the dealer draws several aces, the first could be 11 (if that total would not bust) or could be 1, but the rest of the aces must count for 1.

Input: The first line (N) consists of the number of data sets. Each data set consists of 10 cards.

Output: Print out the hand value of each data set, or “bust” if over 21.

**Example Input file**

6

2 9 A K J 9 8 2 4 10

A 9 A K J 2 4 6 8 10

2 4 6 8 10 J Q K A 3

10 6 5 J Q K A 9 7 4

10 7 5 J Q K A 9 7 4

A 4 A K Q J 10 9 8 7

**(Continued on next page…)**

**(Problem 4 continued)**

**Output to screen:**

bust

20

20

21

17

bust

Explanation of test cases:

In the 1st data set, 2+9=11, so the ace=1, 2+9+1=12 and the king busts at 22!

In the 2nd data set, 11 + 9 =20, dealer holds.

In the 3rd data set, 2+4+6+8 = 20.

In the 4th data set, 10 + 6=16, so draw one more 16+5=21!

In the 5th data set, 10+7=17, dealer holds.

In the 6th data set, 11+4 =15 so the next ace must be 1, total=16, so draw one more and bust!

5. Drought

Program Name: Drought.java Input File: drought.dat

Much of Texas has been under drought conditions for the last 4-5 years. We need rain! Write a program to keep track of monthly rain totals. Given an average yearly rainfall amount and 24 months of data, indicate if the drought is continuing or if the weather pattern is improving.

For this program:

* The drought is over if both years’ data shows at least twice the average.
* The weather pattern is improving only if both years’ rainfall is at least average.
* The drought is continuing otherwise.

Input: The first line (N) consists of the number of data sets. Each data set contains 25 (double type) numbers on one line. The first is the average yearly rainfall, followed by 24 months of rainfall data.

Output: For each data set, print out “drought over”, “improving”, or “continuing.”

Constraints:

1<=N<=10

**Example Input file**

3

**21 1.2 2.0 0.0 1.0 1.0 1.5 0.5 0.6 0.8 0.1 0.0 0.0 1.0 2.0 1.0 2.0 2.5 3.5 3.0 3.0 2.0 2.5 3.0 3.0**

21 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0

**22.5 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8 3.8**

**Note: data file has 3 data sets, each on one line, for a total of 4 lines of input (the 2nd and 4th lines are bold to show this).**

**Output to screen:**

continuing

improving

drought over

6. Cross

Program Name: Cross.java Input File: cross.dat

Given 2 words, you will determine if they can be perpendicular in a crossword puzzle. If they can, you will show them horizontally and vertically intersecting. The first word will be the “across” and the second word will be the “down” word. Here are some examples:

cross and green (the r’s meet)

playing and dog (the g’s meet)

g d

cross o

e playing

e

n

The intersecting letter will be the first letter they have in common (the first common letter in the first word). If there is no common letter, print out “none.” The maximum word length will be 15.

Input: The first line (N) will contain the number of data sets. Each data set consists of two words separated by “and.” The longest word will be 15 characters long.

Output: For each data set, print out either “none” or the intersecting words. Separate data sets with a blank line.

Constraints:

1<=N<=10

**Example Input file**

4

cross and green

playing and dog

meet and non

school and love

**(Continued on next page…)**

**(Problem 6 continued)**

**Output to screen:**

g

cross

e

e

n

d

o

playing

none

l

school

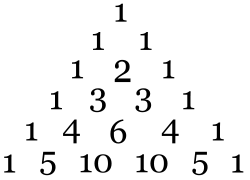
v

e

7. Pascal

Program Name: Pascal.java Input File: pascal.dat

Pascal’s Triangle is a diagram showing the coefficients of a binomial expansion:



(x+1)0 = 1

(x+1)1 = x+1

(x+1)2 = x2 +2x + 1

(x+1)3 = x3 + 3x2 + 3x + 1

Using Pascal’s Triangle, you can find the coefficient of a polynomial of (x+1)y. For this program, find the coefficients of any polynomial of degree 30 or less.

For example, for Y=2, there are 3 values of Z (Z=0,1,2) corresponding to the coefficients 1 2 1 (x2 +2x + 1).

Input: The first line (N) consists of the number of data sets in the file. Each data set consists of a two integers (Y and Z), where Y is the power of the binomial expansion and Z is the position of the polynomial coefficient in expanded form

Output: Print out the coefficient of the polynomial corresponding to Z in each data set.

Constraints:

1<=N<=10

1<=Y<=30

0<=Z<=Y

**Example Input file**

4

2 0

2 1

5 2

10 5

**Output to screen:**

1

2

10

252

8. GPA1

Program Name: GPA1.java Input File: gpa1.dat

In some high school classes, tests do not count a very high percentage of your final grade. In some college courses, however, most of your grade comes from exams. In this program, you have 3 tests and a final. Each test counts for 20% and the final exam is 40%. Find the final average rounded to the nearest point.

Input: The first line (N) consists of the number of data sets in the file. Each data set consists of a four integers, three exams and a final.

Output: For each data set output the final grade.

Constraints:

1 <= N <= 10

**Example Input file**

3

90 90 90 100

100 100 100 0

100 90 80 70

**Output to screen:**

94

60

82

9. Buoyancy

Program Name: Buoyancy.java Input File: buoyancy.dat

Some schools have an “egg-drop” project. The task is to build a device to protect a raw egg when dropped from a certain height. There are different rules and regulations depending on the school and age group. When my daughter had this project, she had a cool idea with helium balloons to hold up the eggs. She was worried that the balloons would lift off! My knowledge of physics helped solve this problem. Using the fact that the buoyant force is equal to the weight of the air displaced, we could calculate how many balloons it would take. The mass of an egg was an average of 55 g, so the weight was 0.54 N.

For each 1.0 L of balloon volume, it would give a buoyant force of 0.011 N. Given the volume of balloons, determine how many balloons it would take to “float” the egg.

Input: The first line contains the number of data sets (N). Each subsequent line contains an integer, Y, the balloon size in liters.

Output: Print out the number of balloons required.

Constraints:

1<=N<=10

**Example Input file**

3

1

5

6

**Output to screen:**

50

10

9

10. Between

Program Name: Between.java Input File: between.dat

Some words have mini-words inside them. For example, the word “attends” has the word “ten” if you take off the a-s and t-d (first-last letter removed each time). In this program, print out each word with the first-last letters removed until the word is 2 or 3 characters long.

Input: The first line contains the number of data sets (N). Each subsequent line contains one word. The maximum word length is 20.

Output: For each data set, print out the original word followed by the smaller words on successive lines. Separate data sets with a blank line.

Constraints:

1<=N<=10

**Example Input file**

3

attends

banana

orange

**Output to screen:**

attends

ttend

ten

banana

anan

na

orange

rang

an

11. GPA2

Program Name: GPA2.java Input File: gpa2.dat

Some high school teachers have a grading policy in which the lowest grade could be dropped. In this program, you can drop either the lowest homework grade (if you forgot to do one) or the lowest test grade (bad day), whichever way will give the student the highest average. The teacher has tests worth 50% and homework 50%, but there could be any number of each type of grade (more than 1 and up to 20).

Input: The first line (N) will contain the number of data sets. Each data set consists of two lines of data. The first line is the list of homework grades. The second line is the list of test grades, each line having at least 2 and at most 20 grades.

Output: For each data set, print out the maximum average rounded to the nearest integer.

Constraints:

1<=N<=10

**Example Input file**

3

100 90 0 70 80

90 80 70

100 90 90 90 88 90 70

90 80 70

100 90 0 70 80

90 50

**Output to screen:**

83

87

79

Explanation of test cases:

In the 1st data set, drop the 0 in hw, so 85% hw and 80% tests is 82.5% = 83%.

In the 2nd data set, drop the 70 in tests, so 88.3% hw and 85% tests is 87%.

In the 3rd data set, dropping the 0 makes 78%, but dropping the 50 makes 79%.

12. Determined2

Program Name: Determined2.java Input File: determined2.dat

I remember in college taking a class called “Linear Algebra” and I thought to myself when I signed up for the class, “Gee, I know lines and linear equations…this should be easy!” Ha.

One important topic in Linear Algebra is a determinant, which some of you in Algebra II class might have seen and used your calculator to type in a matrix and do things with it: find the determinant, use Cramer’s Rule, use inverse matrices to solve systems of equations, multiply matrices of the right dimensions, etc. Even if you haven’t seen this in math class, don’t let that *deter* you from trying this program!

In this program, find the determinant of a square matrix TxT, where T=3,4,5…10. Each matrix element will be an integer -100<x<100.

A 2x2 matrix determinant can be found by:

Given a larger matrix (3x3, 4x4…) it is required to find the smaller 2x2 determinants of a submatrix (minor) using a process called expansion by minors.

By expanding along the top row, cover up each row and column containing element a00 and find the determinant of the minor. Alternately add and subtract the matrix element times minor.

= a00 - a01 + a02

Here is an example of a 3x3 determinant

= - 2 + 3  = 1(-9) -2(-10) + 3(-3) = -9 +20 -9 = 2

The following equation can be used to find the determinant of any general square matrix larger than 2x2:

Where M0i is the minor (submatrix created when removing row 0 and column i) and a0,i is the coefficient element of the matrix row 0 and column i.

**(Continued on next page…)**

**(Problem 12 continued)**

Here is an example of a 4x4 matrix:

= 1( ) – 2( ) + 3 ( ) - 4 ( ) = …

= 1( 18 ) -2 (8 ) + 3 (-12 ) - 4 (-9 ) = 2

You can see why a program is the way to go for this project, especially for matrices larger than 3x3.

Input: The first line (N) consists of the number of data sets. Each data set contains a T by T matrix of integers.

Output: For each data set, print out the determinant.

Constraints:

1<=N<=10

3 <= T <= 10

**Example Input File**

3

1 2 3

2 3 4

5 6 5

1 2 3 4

2 3 4 4

5 6 5 2

8 9 9 4

1 2 3 9 8 7

2 3 4 4 4 3

5 6 5 2 9 8

8 9 9 4 6 5

3 2 1 5 4 3

9 8 7 5 4 5

**Output to screen:**

2

2

408