**A+ Computer Science**

**February 2013 – Packet 1**

Computer Science Competition

Hands-On Programming Set

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.
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.
6. 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.
7. 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.

|  |  |
| --- | --- |
| Number | Name |
| Problem 1 | Picture |
| Problem 2 | Boxes |
| Problem 3 | First Letter |
| Problem 4 | Rocks |
| Problem 5 | Triangles |
| Problem 6 | Hop |
| Problem 7 | Movie Theatre |
| Problem 8 | Reducer |
| Problem 9 | Math Math Revolution |
| Problem 10 | Ganked |
| Problem 11 | Matchmaking |
| Problem 12 | Scheduling |

Good luck!

1. Picture

Program Name: Picture.java Input File: none

**General Statement :** Print out the picture as shown below.

**Input:**  none

**Output:** Print out the picture as shown below.

**Assumptions – Helpful Hints :** none

**Example Input File**

none

**Example Output to screen:**

####################

##<<<<<<<<>>>>>>>>##

#######APLUS########

#######APLUS########

##<<<<<<<<>>>>>>>>##

#######APLUS########

#######APLUS########

##<<<<<<<<>>>>>>>>##

#######APLUS########

#######APLUS########

##<<<<<<<<>>>>>>>>##

####################

2. Boxes

Program Name: Boxes.java Input File: boxes.dat

**General Statement :** Print out each box as shown below.

**Input:**  The first line in the data file will indicate the number of data sets to follow. Each data set will contain the size of the box to be printed.

**Output:** Print out each box of the appropriate size as shown below.

**Assumptions – Helpful Hints :** none

**Example Input File**

3

3

5

4

**Example Output to screen:**

@@@

@ @

@@@

@@@@@  
@ @  
@ @

@ @  
@@@@@

@@@@  
@ @  
@ @

@@@@

3. First Letter

Program Name: First.java Input File: first.dat

**General Statement :** Read in a string and print out the first letter of that string length number of times.

**Input:**  The first line in the data file will indicate the number of data sets to follow. Each data set will contain a single string with no spaces.

**Output:** Print out the first letter of the string length number of times.

**Assumptions – Helpful Hints :** none

**Example Input File**

4

mom  
dog  
pumpkin

apple

**Example Output to screen:**

mmm  
ddd  
ppppppp

aaaaa

4. Comp Sci Rocks!!

Program Name: Rocks.java Input File: rocks.dat

**General Statement :** You think Comp Sci Rocks!! and you just like saying that over and over again. Read in a number from the data file and print out Comp Sci Rocks!! that number of times.

**Input:**  The data file will contain an unknown number of lines.

**Output:** Print out the number of lines in the data file.

**Assumptions – Helpful Hints :** none

**Example Input File**

3

2

5

1

**Example Output to screen:**

Comp Sci Rocks!!

Comp Sci Rocks!!

Comp Sci Rocks!!

Comp Sci Rocks!!

Comp Sci Rocks!!

Comp Sci Rocks!!

Comp Sci Rocks!!

Comp Sci Rocks!!

5. Triangles

Program Name: Triangles.java Input File: triangles.dat

**General Statement :** Read in a letter and a number. The number indicates how big the letter triangle should be. The number indicating the size of the triangle will have a range from 0 to 250. num>=0 and num<=250

**Input:**  The first number indicates the number of data sets to follow. Each data set will contain one letter and one number. All letter input will be uppercase.

**Output:** Print out the appropriate sized letter triangle as shown below. The letter provided is the starting letter and that letter is printed once and then the sequence continues.

**Assumptions – Helpful Hints :** The letters must wrap around from Z to A. If you start with Z and have to print 5 levels, you must wrap around and start with A after the Z level is complete. The triangle lettering starts at the bottom and goes up.

**Example Input File**

3

5 A

3 Z

4 C

**Example Output to screen:**

EEEEE  
 DDDD  
 CCC  
 BB  
 A

BBB  
 AA  
 Z

FFFF  
 EEE

DD

C

6. HOP

Program Name: Hop.java Input File: none

**General Statement :** Print out the word Hop as shown below.

**Input:**  none

**Output:** Print out the word Hop as shown below.

**Assumptions – Helpful Hints :** none

**Example Input File**

none

**Example Output to screen:**

# # #### ####

# # # # # #

#### # # ####

# # # # #

# # #### #

7. Movie Theatre

Program Name: Theatre.java Input File: theatre.dat

**General Statement :** Artemis and Combat Girl love to watch movies. One day they decided to head to Bullet Gorge theatres for a showing of Super Crossfire. This particular theatre has a single row of N seats. Artemis and Combat Girl wants to bring K patrons to the theatre (including themselves). Wascot heard about this event and decides to be a party pooper and enforce that no one can sit next to anyone else. In other words, between every patron, there must be at least one empty seat between them. Wascot is curious about how many possible combinations of seating arrangements there can be given this constraint. Also it doesn’t matter who is sitting in which seat. It only matters the shape of the seating. For example, if Artemis sits in seat 1 and Combat Girl sits in seat 2, this is the same configuration as if Artemis was sitting in seat 2 and Combat Girl was sitting in seat 1. You should not count this twice.

**Input**

The first line of data contains n, the number of inputs. Each input is two integer N and K, the number of seats in the row and the number of patrons in the theatre. N and K will both be less than 12.

# **Output**

Print out the number of valid configurations.

**Example Input File**

3

3 2

4 2

2 2

**Example Output To Screen**

1

3

0

8. Reducer

Program Name: Reducer.java Input File: reducer.dat

**General Statement :** “I’m not from the old-school. I’m from the school that built the old school” is Veteran’s favorite taunt. He likes to use his Ka-Claw to cheese whoever my happen to be standing in the right place at the right time. It doesn’t hurt that he has three types of grapples each of which can reduce the opponent rather quickly. Veteran wants to know given a fraction in the form of P/Q, whats the minimum number of reductions needed to get this fraction in its simplest form. Now of course since Veteran has only a limited set of grapples, you may only reduce with prime numbers. A reduction is defined as dividing P and Q by some prime.

**Input**

The first line in the data file is an integer that represents the number of data sets to follow. Each input contains two positive integers P, Q. Both integers will be less than 1,000.

**Output**

For each input print the minimum number of reductions.

**Example Input File**

3

12 20

9 27

15 14

**Example Output To Screen**

2

2

0

9. Math Math Revolution

Program Name: Math.java Input File: math.dat

**General Statement :** Karl and Assault are Math Math Revolution aficionados. However according to Karl’s calculations, some of these math expressions in the game are not well-formed. “Thus far, events beyond our control, will or intention have been in direct opposition to desire of results” Karl cried. That’s when Assault flew over and exclaimed “Never Fear! Assault is Here!”. You see, Assault has quite the acumen for parsing math expressions. Here are the following rules for a well-formed math expression:

Let X be a well-formed math expression.

1. A positive integer is well formed
2. X op X where op is a binary operator (+, -, \*, /) is well formed
3. (X) is well formed

Note that since Karl and Assault are only playing the Beginner level of the game, the parenthesis is not nested so, for any expression there are at most one set of parenthesis.

**Input**

The first line in the data file is an integer that represents the number of data sets to follow. Each data set is a single line containing one math expression. Each line will have at most 30 characters. There are no whitespaces in the input. The input will only contain the numeric digits 0-9, binary operators, and parenthesis.

**Output**

Print Yes if the expression is well formed and No otherwise.

**Example Input File**

4

85

(5/3)

(5++3)

(5+3+)

**Example Output To Screen**

Yes

Yes

No

No

10. Ganked

Program Name: Ganked.java Input File: ganked.dat

**General Statement :** Captain Spark is highly notorious for his ganking annoyingness. They are highly mobile commandos with the ability to jump into the fray of action, deal massive headache, and get the heck out of there all within a matter of seconds. However these days Captain Spark isn’t too confident about his own abilities. You can often hear him complaining in the jungle. “Why can't they teach us something practical? Like wall-crawling, or pheremone manipulation?” Suppose Captain Spark is in the middle of a ganking attempt, he wants to calculate whether or not he can get out of there. Captain Spark is located at the position marked by a C, empty tiles are marked as ‘E’ and members of the opposing team that obstructs Captain Spark’s path is labeled as ‘X’. Captain Spark can only escape by moving in the 4 orthogonal directions, North, West, South East. It just happens that one of your allies has Slow Roll equipped so all of Captain Spark’s enemies are frozen in place and can’t move. Captain Spark is considered to have escaped if he can reach the point on the map labeled as “Z”.

**Input**

The first line in the data file is an integer that represents the number of data sets to follow. Each data set starts with a line containing two integers R and C, the number of rows and columns of the map. R and C will be less than or equal to 10. The following R lines will each contain C characters representing the map.

**Output**

For each input print Arc Flash if Captain Spark can escape and print Feed if he can’t.

**Example Input File**

2

3 4

CEEE

XXXE

ZEEE

5 5

ZEEEE

EXXXE

EXCXE

EXXXE

EEEEE

**Example Output To Screen**

Arc Flash

Feed

11. Matchmaking

Program Name: Match.java Input File: match.dat

**General Statement :** Matchmaking is a huge problem in Solo Queues. Each game consists of 10 players total, 5 on each team. Each player has some skill level associated with them represented by an integer between 1 and 1000. The goal of matchmaking is to split these 10 players into two teams of 5 such that the difference between the combined skill level of the teams are minimal. If a team consists of players with skill levels S1, S2, S3, S4, S5, the combined skill level of this team is defined as the sum, S1 + S2 + S3 + S4 + S5.

**Input**

The first line in the data file is an integer that represents the number of data sets to follow. Each data set consists of a single line containing 10 positive integers, the levels of the players. Every integer is less than or equal to 1000.

**Output**

For each input, print the minimal difference in skill level between the optimal team assignment.

**Example Input File**

2

5 7 9 3 10 11 5 4 3 8

1 100 1000 9 99 999 5 555 5555 1234

**Example Output To Screen**

1

1781

12. Scheduling

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

**General Statement :** Assassin doesn’t talk much. In fact most of us don’t even realize that she’s female since she’s always wearing that mask. Her comrades lovingly call her by her nickname Sin. “Ugh! Sin, you are such a noob! All you know how to do is Cheese Cheese Cheese.” Or “Arg! Sin needs to be nerfed. She is so OP that it takes no skill to play her”. These phrases of endearments are probably due to Sin’s hard working and persistent personality. During an average day, she would like to backstab N overreaching Hot Shots. She may only backstab a Hot Shot when he is overreaching his boundaries. Each Hot Shot will overreach exactly once during a match for some non-zero interval of time. Obviously no matter how amazing Sin’s mobility is, she can’t be at two places at the same time. Thus she cannot backstab two HotShots if their intervals overlap. Furthermore you can assume that all starting and ending times of intervals are distinct, meaning there won’t be a case where a Hot Shot ends his interval at the same time another Hot Shot starts his interval. Obviously, Sin has to pick her targets carefully, so she would like to know the maximum number of enemies she can backstab.

**Input**

The first line in the data file is an integer that represents the number of data sets to follow. Each input is a single line. The first integer in this line is N (N <= 20), indicating the number of Hot Shots. This is followed by 2N integers in the form of S1, E1, S2, E2,… SN, EN. Where Si and Ei are the starting and ending point of the interval for the i-th Hot Shot. Si < Ei and Si, Ei <= 1,000,000.

**Output**

For each input print the maximum number of Hot Shots Sin can backstab.

**Example Input File**

2

3 5 9 4 12 10 15

3 1 10 2 11 3 12

**Example Output To Screen**

2

1