University Interscholastic League

Computer Science Competition

2014 District Week 1 Programming

JUDGES PACKET AND EDITORIALS - CONFIDENTIAL

I. Instructions

1. The attached printouts of the judge test data and editorial explanations are provided for the reference of the contest director and programming judges. Additional copies may be made if needed for this purpose.

2. This packet must remain CONFIDENTIAL. Additional copies may be made and returned to schools when other confidential contest material is returned.

II. Table of Contents

1. Bank

Program Name: Bank.java   Input File: bank.dat

```
==> bank.dat <==
11
Stanley worked for a company in a big building as employee number 427.
Employee 427's job was simple: he sat in room 427 and worked at his desk.
Stanley was happy.
That coffee will cost you $4.20.
Linux L1 3.5.0-45-generic #68-Ubuntu SMP Mon Dec 2 21:58:52 x86_64 GNU/Linux
2,014 CE
PI, the ratio 4.23 of the circumference of a circle to its diameter, or
3.1415 926 5,35 - and this  is just the beginning. It keeps on going,
forever, without 1 2 3 4 ever repeating, which means that contained within
this string 4.3 of decim 09 als is every 03.020 single other number
:D

==> bank.out <==
427
427
427 427
854
-
0
4 20
24
1 3 5 0 45 68 2 21 58 52 86 64
405
2 14
16
4 23
27
3 1415 926 5 35
2384
1 2 3 4
10
4 3 9 3 20
39
-
0
```

2. Code

Program Name: Code.java   Input File: code.dat

```
==> code.dat <==
5
zikhmpxjfsvqdaolywtecrbugn 1
2
E UIL Computer Science rocks
D dgr 2014 xplwu
xyikbnmpvaqouctdjszerwlhfg -2
3
E sup bro!
E abcdefg
D 3947
mlnpvjbexoqhzkdgawusrcitfy -18
5
D jeufs
E aft
D efvd
E boat
E cheer
gcnjwsxvlheyztpqfkiumbdaro 23
4
D rhdiq
D amhg
E feelings
D qmhrs nmvt
jgbucsihrmpaqefoyvdlkntzxw -21
10
D tfowj
E animal
E operand
E setting
D nhjqwf
E draper
D lhnwfz
E reaggravation
E anteorbital
D fhotwjn
```

```
==> code.out <==
Protocol 1
UIL Cpemdfnx Slgnbln xplwu
uil 2014 rocks

Protocol 2
xpb wqr!
vwgizlk
3947

Protocol 3
grain
ura
rice
tlua
vmdde

Protocol 4
train
word
pttviquf
north pole

Protocol 5
plain
ojwvof
kthaojz
ihqqwjn
gentil
zaotha
begild
ahonnaosoqwkj
ojqhkalwqof
leaping
```

3. DNA

Program Name: DNA.java   Input File: dna.dat

```
==> dna.dat <==
7
ATGC
TACG
ATGC
CGTA
AGQ
TCF
CGATAGAT
CCTATCTA
CTTGCTTCGGAAGTCCCGGTGGACC
GAACGAAGCCTTCAGGGCCACCTGG
GGTCGTATCGCT
CCAGCATAGCGG
GTATTAAGGATCCAATTGGTTTCCAATGTATTGAAGCGTCCTCCGGCCATACTCAAGGACGCTGTTAAG
CATAATTCCTAGGTTAACCAAAGGTTACATAACTTCGCAGGAGGCCGGTATGAGTTCCTGCGACAATTC
```

```
==> dna.out <==
GOOD
BAD
BAD
BAD
GOOD
BAD
GOOD
```

4. Exponentiation

Program Name: Exponentiation.java   Input File: exponentiation.dat

```
==> exponentiation.dat <==
10
5.0 1.0 2
4.5 -7.5 3
-1.0 10 5
2.0 3.0 1
2.0 3.5 2
2.5 4.5 3
0.0 2.0 1
0.1 2.0 3
0.25 2.0 4
3.0 2.0 4

==> exponentiation.out <==
(24.0,10.0)
(-668.25,-33.75)
(-49001.0,90050.0)
(2.0,3.0)
(-8.25,14.0)
(-136.25,-6.75)
(0.0,2.0)
(-1.199,-7.94)
(14.50390625,-7.875)
(-119.0,120.0)
```

5. Fillings

Program Name: Fillings.java   Input File: fillings.dat

```
==> fillings.dat <==
10
2 3
2 10
4 9
5 7
3 6
3 9
5 8
3 4
10 10
1 2

==> fillings.out <==
6
90
3024
2520
120
504
6720
24
3628800
2
```

6. JSON

Program Name: Json.java   Input File: json.dat

```
==> json.dat <==
17
Cake.ingredient = sugar
print(Cake)
Cake.bake = oven
print(Cake)
dog.treat = bone
print(dog)
boat.type = sailboat
boat.length = 33
boat.color = white
boat.year = 1968
print(boat)
uil.topic = CS
uil.test = 40_MC
uil.packet = 12
uil.topScore = 1440
uil.status = CS_Rocks!!
print(uil)

==> json.out <==
{ingredient : sugar}
{ingredient : sugar, bake : oven}
{treat : bone}
{type : sailboat, length : 33, color : white, year : 1968}
{topic : CS, test : 40_MC, packet : 12, topScore : 1440, status : CS_Rocks!!}
```

7. M&M's

Program Name: MandMs.java   Input File: mandms.dat

```
==> mandms.dat <==
10
1
2
3
5
100
6
7
8
9
101

==> mandms.out <==
1
2
3
4
3
3
4
3
4
4
```

8. Rabbits

Program Name: Rabbits.java   Input File: rabbits.dat

```
==> rabbits.dat <==
10
1
2
3
5
14
6
8
30
15
115

==> rabbits.out <==
1
1
2
4
129
6
13
58425
189
7536815746437618530
```

9. Right Hand

Program Name: RightHand.java   Input File: righthand.dat

```
==> righthand.dat <==
6
3
...
.##
.#.
5
.###.
...#.
.###.
...##
.#...
10
.....#..##
#.#..###.#
#...#.#...
.#.##.##..
....#..##.
..#....#..
.####.#.#.
#.#.#.##..
.........#
.#.....#..
5
.....
...#.
.###.
#..##
.#...
20
..#....####......#..
#..#..#.#..#.#...#..
#.##...#.##...#...##
....###...#.##.##..#
...#...##...#...##..
......#..#....#.....
..#.##..#.###..#....
#...##......##...#..
...##.....##..#....#
#..##..#.##..#...#..
.#.......#...##.....
.....##.#..##.#.#.#.
#.#.#..#.##...##...#
#.##.#..............
#.#..##....#........
..#...##.#........##
#.#...#.#.#....#....
#..##....##...###..#
.......###..#.....#.
###...#.#.###.#.##..
25
...#...#...##...#..#.#..#
#..#..#.#...##..##..##...
.#..#.........#.......##.
...#..#.....##.#.####....
##......#..##.#....#..##.
...#...###...###.#..#....
##.####..#.#...####.....#
.#....#....#..#..###.....
####.#.....#.#.....#...#.
#.#....#..#...###....#.#.
#..#...####.#.....#....##
....#.###.###.......##.##
#.#.#....#.#...####.....#
#....#..#..#..#.......#.#
..#....#...#.##....#.##..
..#.####.....#..##......#
......##...#....#........
.....#...#..###....#.#...
##...#.#........#.##...##
###......#..#....##.#..#.
.##..............##.#.#.#
.#..#.###.#.###...#.....#
..##...####...#...#.#....
...#......#.######..#.#..
#.#...#...#........##..#.
```

```
==> righthand.out <==
NO
YES
YES
NO
NO
YES
```

10. Stealing Gold

Program Name: Stealing.java   Input File: stealing.dat

```
==> stealing.dat <==
10
3 2 10
35 68 42
25 70 1
63 59 79
46 6 65
62 28 82
43 96 92
92 37 28
54 3 5
22 83 93

==> stealing.out <==
30
42
1
237
975
410
92
140
175
93
```

11. Tale of a Tail

Program Name: Tale.java   Input File: none

```
==> tale.out <==
This is a
 very long
  and un-
 happy
tale
  .
```

12. Teams

Program Name: Teams.java   Input File: teams.dat

```
==> teams.dat <==
20
11 98
16 149
31 116
12 40
53 14
23 187
14 163
27 21
26 31
19 42
18 56
21 1
20 171
22 85
15 171
17 192
25 49
24 84
29 191
28 34

==> teams.out <==
98
51
131
127
157
30
133
112
455
123
179
178
349
264
435
285
578
494
305
271
```

University Interscholastic League
Computer Science Competition

2014 District Week 1 Programming
Judges Packet – Editorials

CONFIDENTIAL

Table of Contents

1. Bank

Process the text line by line to identify numbers. You may use the sample solution's methods or Character.isDigit() etc.

2. Code

Overview of problem:
We have to manipulate a string character by character according to a given function. We have to look up the character in a map to do the substitution, then math on the ascii value to produce the correct rotation.
Analysis of problem:
Since this translation creates a one-to-one mapping of characters, if we figure out the mapping for encoding, we can create a reverse mapping for decoding without actually having to calculate anything.
Solution:
For each protocol, we first construct the encoding mapping using the substitution string and the offset. Then we can construct a decoding mapping by reversing the encoding mapping. Then, for each phrase we need to encode or decode, we just substitute each character in the string by the one it maps to in the specified mapping.
Constructing the mapping is O(26) for each mapping, and doing the encoding/decoding is O(length of string to encode or decode).

3. DNA

Overview of problem:
We have to compare two strings at each position to see if they satisfy certain properties.
Analysis of problem:
The requirement for each string being "good" is that at each index, the two characters from the strings are one of the four pairs.
Solution:
Simply iterate through the string, and at each position, evaluate whether those two characters are one of the valid pairs. If all of the pairs are valid, we say the pair of strings is good, but if we find any that aren't we say that the pair of strings is bad. This solution runs in O(n)time, where n is the length of the string.

4. Exponentiation

Solution uses recursion and given formulas.

5. Fillings

Problem explanation:

Let's see a simpler analogy of this problem. There are N slots and there are integers from 1 to M which are to be filled in the slots (one in each slot) such that all slots have distinct integers.

Solution:
First slot can be filled in M ways.
Second slot cannot be filled with the same integer as the first one. Hence, it can be filled in M – 1 ways.
The answer is: M*(M-1)*(M-2)…*(M – N + 1)

6. JSON

The solution uses a Map to map between the object name and a list of values.

7. M&M's

Let us say he starts with N M&Ms.
The number eaten on the last turn that leaves nothing behind has to be 1 since he cannot put all the remaining pieces in a single row and eat them all, nor can he arrange them in a rectangle and eat all the rows. The only way to get to 1 piece left behind is if he reduced it from 2 by eating one.
If he starts off with an even number i.e. N is even, he can get to 2 in one more turn by arranging the piece in a rectangle 2 x N/2, and eating all but one row -- i.e. N-2 pieces. Thus if N is even and > 2, he can finish off the pieces in 3 turns.
If N is 3, he has to reduce number to 2, 1, and 0 successively in 3 turns.
If N is odd and > 3, the fastest way to reduce the number of pieces to 2 is to eat one and leave behind an even number, since he cannot arrange an odd number of pieces in a rectangle of width 2, as 2 is not a factor of any odd number. So an odd number > 3 will take 4 turns.

8. Rabbits

Overview of problem:
We first need to derive the formula and then write an efficient program to calculate values of the formula.
Analysis of problem:
In deriving the formula, the key insight is that rabbits in this model take one more month before they produce offspring than in the original model. The formula f(n) = f(n-1) + f(n-2) is derived from the fact that all of the pairs of rabbits from the previous month live on, and all of the pairs that were there for one month already (the number of pairs at n-2) produce a new pair this month. Thus the formula is f(n) = rabbits from last month + new rabbits = f(n-1) + f(n-2).
With the insight into the new model, we see that the rabbits that reproduce are not from month n-2, but month n-3 instead. Thus, the simple modification gives us the new formula f(n) = f(n-1) + f(n-3).
Solution:
Defining a function similar to this one:

```
public int f(n) {
   if (n <= 2) return 1;
   return f(n-1) + f(n-3);
}
```

Is not going to be efficient at all. Looking at the call stack, for f(3), we get f(3) = f(2) + f(0) = 1 + 1
For f(4), we get f(4) = f(3) + f(1) = f(2) + f(0) + f(1) = 1 + 1 + 1
For f(10), f(10) = f(9) + f(7) = f(8) + f(6) + f(6) + f(4) = f(8) + f(6) + f(6) + f(4) + …..
As you can see, the number of function calls grows very fast. In fact, f(n) will be called more times than the value of f(n)
Thus, for your upper limit of f(115), we have f(115) = 7536815746437618530. This number is 19 digits, and given that it will make significantly more function calls than that, this is clearly not an approach that can be expected to complete within the time allotted for the contest.
Thankfully, there is an efficient solution. To compute f(n), we need to know the f(n-1) and f(n-3). More specifically, to compute f(n), we need to keep track of the previous 3 values of f(n). This gives us an incremental approach, where we calculate f(i) based on the previous 3 values, throw out f(i - 3) since it is no longer needed, and store f(i). Then we increment i up until it reaches the value we need, or n.
This approach is O(n) time instead of exponential time, so given a max n of 115, this will clearly be fast enough.
Additional note: since f(115) is greater than 2^31 – 1 ,it will not fit inside a 32 bit integer, so we need to use longs (64 bit integers) to hold the result of the computations.

9. Right Hand

You are told about the Right-Hand rule, which applies to maze, and you are asked to determine if a maze is solvable with the right-hand rule.

The trick to notice is that if a maze is solvable by the right-hand rule, then the maze is solvable with any valid solution. If it's not solvable with the right hand rule, then the maze is impossible to complete in any case. Therefore, the question is simply asking you to determine if the maze is solvable starting from the top left corner going to the bottom right corner.

Therefore, the solution provided uses a Breadth First Search (BFS) to find if there exists such a path that starts at the top left and ends in the bottom right. If it finds such a path, then we print Yes, otherwise, we print No.

10. Stealing Gold

Solution:

At one time, the person can carry M units of gold. He goes from Bank to the Car and goes back to the Bank only if the gold is back.
If N is divisible by M, then total time will be (N/M)*2*T - T as he need not go to the back one last time.
Else the total time will be (N/M)*2*T + T, as he has to get the remaining N%M units from the bank to the car.

11. Tale of a Tail

Simple output.

12. Teams

See explanation of example testcase, reproduced below.

Example Input File

```
5
2 3
1 7
5 5
3 1
8 15
```

Example Output to Screen

```
3
4
5
4
9
```

Explanation of Sample Case:
After the first person shows up, he joins the young team, since the young team gets the advantage when there are an odd number of contestants. Thus, the total rating on the young team is 3, and the total rating on the old team is 0. Then the next contestant shows up. Now there are two, and since (1, 7) is younger than (2, 3), the young team is now (1, 7) and the old team is now (2, 3), so the difference in their rating is 4. After the third person,(5, 5) shows up, the young team is (1, 7) and (2, 3), and the old team is (5, 5) so the difference in their ratings is 7 + 3 – 5 = 5. The fourth person shows up, the teams are (1, 7) and (2, 3) vs (3, 1) and (5, 5), so now the difference is 4. The fifth and final person shows up, now the teams are (1, 7), (2, 3) and (3, 1) vs (5, 5) and (8, 15). These add up to ratings of 11 vs 20, so the difference is 9.