Computer Science Contest #1415-13 Key

February 14, 2015

1) E (B and D)

2) C

3) E

4) D

5) E

6) A

7) C

8) C

9) D

10) A

11) B

12) D

13) C

14) A

15) D

16) C

17) B

18) D

19) C

20) A

21) E

22) C

23) D

24) E

25) B

26) E

27) A

28) C

29) B

30) D

31) B

32) C

33) D

34) B

35) E

36) D

37) C

38) B

39) 7 or seven

40) 30

Note to Graders:

- All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g. error is an answer). **Ignore any typographical errors.**
- Any necessary Standard Java 2 Packages are assumed to have been imported as needed.
- Assume any undefined (undeclared) variables have been defined as used.

Brief Explanations:

1. $41_{16} + 31_{10} + 21_8 + 11_2 = 65_{10} + 31_{10} + 17_{10} + 3_{10} = 116_{10} = 164_8$
2. Order of operations is changed with use of parentheses: 17 * (17-15) = 17 * (2) = 34
3. The %.3d should be %.3f   The runtime error message reads, "IllegalFormatPrecisionException".
4. If p and q are both false then (p && q) = false and (p||q) = false resulting in false when they are compared using OR. Any other combination will result in true.
5. s is an object therefore a null value will cause a NullPointerException. The null String value: "" would result in a response of true.
6. The random generator is generating one of 50,51,52,53, or 54 for each of the 5 iterations. 55 will never be generated.
7. When using assignment operators everything on the right side is done first, like putting parentheses around them: x = x * (y + z * x)
8. The ASCII value for 't' is greater than the ASCII value for 'o' so substring(4,7)   "log"   is printed. The last line of the code, though indented incorrectly, will execute independently of the if/else statement every time the code is run printing "at".
9. The values generated by the loop are 2, 5, 8, 11, and 14 so one is added to each of these values as they are printed: 3691215
10. The variable **sum** represents a count of how many numbers in the array have a remainder of 2 when divided by 3.
11. When entering a string value into a program you do not need to enclose it inside quote marks. When counting the length be sure and include spaces and punctuation. When counting for location of characters be sure and start with zero and include the spaces and punctuation again.
12. Loops through the numbers from 23 to 30 and assigns them to i which is a double. The division by 3 is now double division resulting in the final sum print formatted to **70.6667**
13. The += and -= have lower precedence than * and – and + so the right side is done first inside each inner parentheses. On the left side of the && the equation is != to 0 so the right side is checked. Once the math is done inside the parentheses then / has a higher precedence than > so the division is done first and now the result can be compared to 15. It is worth noting that both these sets of inner parentheses must be present else you will get an "unexpected data type" for the division symbol. The problem boils down to the last check: k-=3-2 Since k was assigned a new value (1) during the execution of the left side of the && k now equals zero and you get a runtime error – division by zero.
14. There are an even number of integers total so think of it this way. If you have a 1000 numbers to divide equally between negative and positive, your positive numbers would go 0 to 499 which is 500 numbers counting the zero. The negative numbers would go -1 to -500. Add -500 and +499 you get -1.
15. ArrayList commands: **add** with one number adds the number to the end of the list, **add** with two numbers inserts the second number at the spot designated by the first number bumping everything to the right. The set command changes whatever is in the location designated by the first number to whatever the second number is. The loop prints the values front to back.
16. Accessor methods retrieve information without changing it. There are 2 accessor methods: getTotals() and toString()
17. The <1> print command prints: **Name: Shenzu   Faction: undecided** (with a lower case u which was entered into the changeFaction( ) parameter).
18. The correct answer is D. There were 3 Horde and 1 Alliance entered originally. One of the Horde was changed to Uncommited so - 2 1 1
19. No out of bounds…trace the values in: 00000 22220 44420 66420
20. The result of right-shifting a negative number by N will be the same as dividing by 2^N, rounding toward negative infinity
21. Inheritance – The super call in Two's default constructor assigned Aurora to s in class One. Variable s in class Two has never had anything assigned to it so its value is null.
22. Inheritance – Two has an invisible call to One's default constructor supplied by Java so s in class One is set to "name". The c.getName() call retrieves the value from One's s variable because the getName( ) method was inherited from One.
23. Inheritance – b is now a reference to a One object.
24. Inheritance – since b still refers to a, it is limited to the One class and no longer knows about the setName( ) method in the Two class.
25. Last even number - 18%2=0   nums[18/2] = 31 (This is not a running total question)
26. A search in a stack returns the distance of the location of the object from the top of the stack. The top item in a stack is indexed with 1 not 0.
27. itisallamystery
28. m(7) = 7 + m(5) + m(4) = 7 + 16 + 11 = 34

    m(5) = 5 + m(3) + m(2) = 5 + 7 + 4 = 16

    m(4) = 4 + m(2) + m(1) = 4 + 4 + 3 = 11

    m(3) = 3 + m(1) + 1 = 3 + 3 + 1 = 7

    m(2) = 2 + 1 + 1   = 4

    m(1) = 1 + 1 + 1   = 3

29. f(7,4) = 7 + f(6,4) = 7 + 21 = 28

    f(6,4) = 6 + f(5,4) = 6 + 15 = 21

    f(5,4) = 5 + f(4,4) = 5 + 10 = 15

    f(4,4) = 4 + f(2,4) = 4 + 6 = 10

    f(2,4) = 2 + 4   = 6
30. binary *and* operation + binary *exclusive or* operation 7 + 6 = 13
31. Shortcut: determine the first power of 2 that is >= the number of elements in the array. In this case 2^{10} so the answer is 10 (2 4 8 16 32 64 128 256 512 1024) 🡨 count the numbers
32. When the array was loaded the last cell was left at 0. The TreeSet moves it to the front.
33. 13 - Split on any individual occurrence of the letters a through e, a space, and the digits 0 through 6   p/*/rl/h/r/or/*/*/*/*/7/*/9//   *=null splits at end (41) are ignored
34. Since all numbers pushed by %3 = = 0 are immediately popped by the following number %3 = = 1 the only thing left besides the sevens is 15, the last number added to the stack that was not popped.
35. A stack is printed out in the order the numbers were pushed onto the stack, left to right, but when peeking or popping it uses the last cell. [7 7 7 7 15]
36. Post-Order Traversal (left-right-data)   CFDKSTURNH

37. 3 – T T T, F T T, F F F

Chart below starts with all possible combinations of A B C

A B C	(A && !B)	B^C	!((A & !B) || (B^C))
T T T	F	F	T **
T F T	T	T	F
F T T	F	F	T **
F F T	F	T	F
T T F	F	T	F
T F F	T	F	F
F T F	F	T	F
F F F	F	F	T **

38. $+ - + \uparrow X\ 3 * 3 \uparrow X\ 3 * 3\ X\ 3 = + - + (\uparrow X\ 3) * 3 (\uparrow X\ 3)(* 3\ X)\ 3$

$= + - + X^3 (* 3\ X^3) (3*X)3 = + -(+ X^3 (3*X^2)) (3*X)3$

$= + -((X^3 + 3*X^3) (3*X))3 = + (X^3 + 3*X^3-3*X)3$

$= X^3 + 3*X^3 -3*X + 3$

$X = 2 \Rightarrow 2^3 + 3 * 2^3 - 3 * 2 + 3 = 8 + 24 – 6 + 3 =$ **29**

39. You can solve this problem mathematically by representing the graph in a matrix:

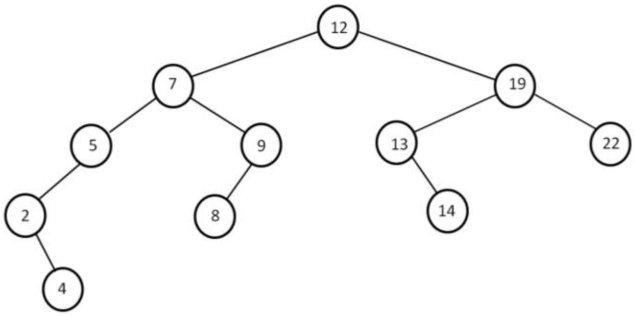
.	A	B	C	D
A	0	1	1	1
B	0	0	1	0
C	1	1	1	1
D	1	0	0	0

Each 1 represents a single edge of the graph that starts at the vertex in the far left column going to the vertex on the top row. Going from A to A there are 0 edges, from A to B there is one edge, A to C one edge, B to A zero edges, etc. Notice there is an edge going from A to B but not one going from B to A. That is because in this graph this edge only goes one direction. Now that you have the matrix constructed (call it *M*) you need to raise it to the power of the length of path you are looking for, in this case 3. (M^3) After doing the math the resulting matrix looks like this which shows all paths of 3 starting at the vertex on the left and ending at the vertex at the top:

.	A	B	C	D
A	3	4	5	4
B	2	2	3	2
C	5	5	7	5
D	2	1	2	1

The paths for each vertex ending at the same vertex are in the diagonal from the top left corner to the bottom right corner of the matrix. We are only interested in paths that start and end at the C vertex, so check row C at column C and the answer is 7. You can do this visually and you will end up with the following paths: C-A-B-C, C-D-A-C, C-A-C-C, C-B-C-C, C-C-A-C, C-C-B-C and C-C-C-C.

40. After constructing the tree the sum of the longest root-to-leaf path is 12 + 7 + 5 + 2 + 4 = 30


12
7
19
5
9
13
22
2
8
14
4