Computer Science Contest #1718-04 Key

October 28, 2017

1)	E
2)	A
3)	D
4)	D
5)	E
6)	D
7)	C
8)	A
9)	B
10)	A
11)	E
12)	E
13)	B
14)	A
15)	A
16)	C
17)	E
18)	D
19)	C
20)	A
21)	E
22)	A
23)	C
24)	D
25)	A
26)	C
27)	B
28)	C
29)	E
30)	D
31)	A
32)	E
33)	C
34)	D
35)	E
36)	B
37)	B
38)	D
39)	1101 1001
40)	000110

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. 101111_2+101000_2 = 1010111_2 = 1+2+4+16+64 = 87_{10}
2. 34 + 2 / 34 – 2 = 34 + 0 – 2 = 32. (integer division, no decimals)
3. %-10s means that 10 spaces are set aside for the variable which is a String and it will be left aligned to the set aside spaces.
4. substring always looks at the index of the String and not the value, the second index is not included in the new String.
5. AND (&) has a higher precedent than XOR (^), so b&c is solved first, then the xor applies afterwards. In order to be true, both sides must be opposite of each other.
6. ceil() is a Math method that rounds up and returns a floating point decimal.
7. x = 37, y = 37/12 = 3, z = 3%6 = 3, x = 37*(3-3) = 0.
8. 332%5 = 2. it will go to case 2 which also happens to yield the same results.
9. The trick here is to understand you are increasing my 7 while decreasing by 7 as well. The halfway point is 163 which is not divisible by 7, 7 goes into 163 approximately 23. 7 * 23 = 161, this is the last value i will be before the loop ends. But x is working backwards from this so: 326 – 161 = 165. However, you didn't start at i=1, you started at i=0 which means 165-7=158. Or, you could use logic about how loops work and realize that x has to be less than halfway of the max (326/2=163), so that the only feesible answer is 0 or 158. Crunching is not always the fastest way to a solution, but rather use your mathematical skills…computer science is mathematics.
10. indices start at 0. So, it is 23-38=-15.
11. All of them can be used, even hasNextDouble because, although the value is being read in as an int, the value could be read as a double. If we had said hasNextBoolean, then the code would not have worked.
12. 1/4 = 0, so the loop never goes forward and nothing is printed out.
13. the order of precedence can be found at http://introcs.cs.princeton.edu/java/11precedence/
14. ~ is a complement. The complement on 0 is -1, 1 is -2, …, and 5 is -6
15. The list is constantly growing and shoving the previously enterend values to the right when add is called.
16. This is what every professor who teaches computer science java object orientated programming puts as their first question on their exam. Know it, learn it, love it, live it.
17. you must call the super class' chargeBattery because you have no access to the encapsulated charge other than the modifier getBattery (which wasn't an option). So the only possible answer is super.chargeBattery(x/3). Oh look! encapsulation! polymorphism! AND inheritance! What a coincidence.
18. if you remove from a list from the front to the back, you will skip over the next value if you do not immediately decrease the index by one.
19. %13 will yield [0,12]. [0,12]-6 yields [-6,6] which is the range
20. The while will stop the first time it comes across a 0, not the first time it creates a 0.
21. 17 > 5 so n becomes 17/10 = 1. 1 < 5 so x = 4.
22. The unique numbers are [-3,7] created by 0,1,2,3,4,5,6-10,11-20,21-30,31-40,40 on up.
23. 26 = 11010, 31 = 11111, 30 = 11110, 5 = 101
26|31&30|5 = 11010|11111&11110|101 = 11010|11110|101 = 11111 = 31
24. The truth table is an inversion of A||B, so it is !(A||B). If you were to distribute the !, then it would be !A&&!B.
25&26. The method takes a string and puts it into a 3 row matrix of suitable size in column major older (fill the column first).
27. The recursion will return 10+3*(6-x)=10+3(6-0)=10+18=28.
28. The recursion will return 10+3*(6- -(10+3*(6-(-2))))=

    10+3*(6- -34)=10+3*40=10+120=130.
29. A:1010 C:1100 7:0111 = 101011000111
30-32. The height of the tree is the number of edges from the root to the lowest leaf. In this case 3. 32 is on level 3. Post order printing means you print the parent only after you print the left child and the right child. The root will always be the last value printed.

```
    17    level 0
  /    \     1
 2     46 level 1
  \   /      2
  13 34   level 2
    /        3
   32     level 3
```

33. A priority queue must is a min-heap. So all parents must be smaller than its children with the root being the first element of the queue.

```
      1          So, the array is [1,4,5,15,7,48,42]
    /   \
   4     5
  / \   / \
15   7 48 42
```

34. 3274>>3 == 3274/2^3=3274/8=409
35. The first one is in index 0 because the String starts on an I, the second one is created because of the NN part of the String.
36. !AB!(A&&C)+!B(!A!C)=!AB(!A+!C)+!A!B!C=!A!AB+!AB!C+!A!B!C=!AB+!AB!C+!A!B!C=
    !A(B+B!C+!B!C)=!A(B(1+!C)+!B!C)=!A(B+!B!C)
37. NAND is also known as a NOT AND. The little circle means NOT.

```
38. ABC+.5^/.5^1DE-2^/*
    A(B+C).5^/.5^1(D-E)2^/*
    A((B+C)^.5)/.5^1((D-E)^2)/*
    (A/((B+C)^.5)).5^(1/((D-E)^2))*
    ((A/((B+C)^.5))^.5)(1/((D-E)^2))*
    ((A/((B+C)^.5))^.5)*(1/((D-E)^2))
    (A/(B+C)^.5)^.5*1/(D-E)^2
```

39. two’s compliment 39 = 00100111 flip bits 11011000 add 1 11011001 and that’s -39.

```
40.
  NOT (RCIRC 4(00011101) XOR LSHIFT-2(x)) = 00110110
>invert
  RCIRC-4(00011101) XOR LSHIFT-2(x) = 11001001
>simplify RCIRC
  11010001 XOR LSHIFT-2(x) = 11001001
>XOR 11010001
  x = 00011000
>RSHIFT 2
  x = **000110
```