Computer Science Contest #1415-09 Key

January 10, 2015

1) B

2) E

3) B

4) C

5) E

6) E

7) E

8) C

9) E

10) A

11) E

12) E

13) B

14) B

15) D

16) A

17) E

18) D

19) C

20) E

21) A

22) B

23) C

24) D

25) B

26) B

27) E

28) A

29) E

30) A

31) B

32) D

33) B

34) D

35) B

36) D

37) E

38) C

39) -39

40) (0,1,0) (0,0,1)

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.

© A+ Computer Science – www.apluscompsci.com

Brief Explanations:

1. 10001101000_2 + 631_{10} -> 10001101000 + 1001110111 -> 11011011111 -> $6DF_{16}$
2. 21 % 16 + 21 * 4 – 16 -> 5 + 84 – 16 -> 73
3. the - symbol means to left justify.
4. Strings are immutable, so no matter what the replace does, the String stays the same.
5. This is an XOR statement, so a and b must be opposite while c must be true.
6. The ^ sign is the bitwise XOR statement and cannot be used to take the cube root.
7. The values of i will be 25,35,45,55,65,75,85,95->480*2->960->5000-960=4040
8. break tells the switch to stop, so x will be 52+20+37=109.
9. i will be 8,7,6,5,4,3, and 2
10. mid = (8-1)/2 -> 7/2 -> 3. list[mid]=35, list[mid-1]=5, list[mid+2]=37.
11. All four work because it evaluates the hasNextWhatever after the nextInt is done, since the next number could be an int, double, or string.
12. The x[i] goes 1->4->3->3->2->3->2.
13. Conditional AND (&&) must wait for bitwise operations (&, >>), concatinations, and comparisons (<).
14. The short integer is a signed data type so it will be -2^{15}-1 = -32768.
15. Remember when you remove an element the rest of the array moves to the left and the new indexing will change.
16. The attributes must be private when encapsulated.
17. Since str has a higher priority, the only time you would check the clan is if the str is the same.
18. No matter how you go thru this part, it is just summing up the values.
19. The trick here is to know where you are going: i is the col, j is the row:

```
j = row = 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3
i = col = 0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3
```

```
20. 110000 << 2 | 100110 << 1 & 110000 – 100110
    11000000    | 1001100     & 1010
    11000000    | 1000
    11001000 => 128+64+8 = 200.
```

21&22. The index starts at 3 going left and will continue until it hits a zero which coincidentaly is where it started (the third index will be decreased faster than the zeroth index)
23. Engine must be implemented because it is an interface, V8 must be extended because it is a class
24. V9 is an Engine because it inherits from the V8 class.
25. Although it is tempting, Object cannot be placed in <*4> because Object does not have the getFuel method.
26. Since there is no break in the switch statement, every car eventually becomes a V10. There is no division by zero error during this time because getFuel is called in after the car is made a V10.
27. A priority queue is a min heap tree where the parent is smaller than either children. It is printed out in level order left to right.

```
28. f(15)=f(13)+5=13
    f(13)=f(11)+5=8
    f(11)=f(9)+5=3
    f(9)=f(8)-2=-2
    f(8)=f(7)-2=0
    f(7)=f(6)-2=2
    f(6)=f(5)-2=4
    f(5)=6
```

29. you know that f(9) is going to be -2, the difference between -2 and 26 is 28 which is not divisible by 5 so we know that previous method call was not f(11), so it must have been f(10) = -4. The difference between -4 and 26 is 30, which means 6 more calls where x is 12, 14, 16, 18, 20, 22. x must be 22.

30. This is a plug and play question. We can work backwards

```
    112 = 1110000 goal
    ^19 = 0010011 will lead to
          1100011
    ^81 = 1010001 will lead to
        = 0110010
    ^ x = ??????? will lead to
     42 = 0101010 start ---- so x must be 0011000 = 24
```

31. This is a sorted binary search tree, however the values that go left are the greater values of the root and the values that go right are the lesser values of the root. It is printed in pre-order traversal

32. the first time i increases is when 29>22, and it will increase by 2 (popping it off a and then pushing it back on). The next time it will occur is when n is 18 (6 more pushes, because it pops off the entire a stack and places it back on). For a total of 8. By the way, the stack is sorted when the code is done.

33. although contains is tempting, containsKey is the only one that will work.

34. this is a tree set so every team that is linked to 1 will only appear once.

35. k{2} means exactly 2 ks must start the string, .+ means 1 or more characters are next, followed by 3 ks followed by 1 or more characters and ending with a k.

36. !(BC) + A(B+!A) + AC

```
    !B + !C + AB + A!A + AC
    !B + !C + AB + 0 + AC
    !B + !C + A(B+C)
```

Note: the only time this is false is when A=false and B=C=true, which means it can’t simplify down to true.

37. RSHIFT-2(11000 XOR 10011) => RSHIFT-2(01011) => 00010

```
    LSHIFT-1(?) OR 00010 = 00110.
    LSHIFT-1(?) = 001a0. Where a can be anything.
    10110 AND RCIRC-3(x) = a001a.
    RCIRC-3(x) = aa01a
    x = 1aaa0 = 11010
```

38. /*-+ABC--ABC***++ADFCBA

```
    /*-(A+B)C-(A-B)C***+(A+D)FCBA
    /*(A+B-C)(A-B-C)***(A+D+F)CBA
    /((A+B-C)*(A-B-C))**((A+D+F)*C)BA
    /((A+B-C)*(A-B-C))*((A+D+F)*C*B)A
    /((A+B-C)*(A-B-C))((A+D+F)*C*B*A))
    ((A+B-C)*(A-B-C))/((A+D+F)*C*B*A)) – that’s the answer, now clean it
     (A+B-C)*(A-B-C)/(A*B*C*(A+D+F)) – is equivalent, math is our friend
```

39. This is two’s compliment, since it is signed you know the number must be a negative, to find the value of the negative you must invert all the bits and add 1, then find the decimal value.

11011001 => 00100110 + 1 => 00100111 => 39_{10} => -39

40. The digital circuit diagram is !A && (B XOR C). This mean B and C cannot be the same and A must be true. There are only two possibilities that will result in true = (1,0,1) and (1,1,0).