Computer Science Contest #1415-11 Key

January 24, 2015

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

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. $23_6 – 23_4 = 15_{10} – 11_{10} = 4_{10} = 4_5$
2. This is a legal program, in essence the answer is 3 + 4 = 7
3. True or false is true.
4. The variable `one` is not augmented in this code, therefore the output is simply `mander`.
5. Both the arrays are initially filled with 0's, thus 0 + 0 = 0. Variable name has no effect on datatype.
6. ct starts at 1, the loop is iterated 25 times, ct = 26
7. The value `a` has nothing to do with `q` but it still is a factor in the combination count. The following combination of `j`, `k`, `l` results in `q` being set to false. Refer to a Java operator precedence table.

j	k	l	a
false	true	true	false
false	false	true	false
false	false	false	false
false	true	true	true
false	false	true	true
false	false	false	true

8. All values are initially false in a boolean array, and after the loop all the even numbered positions are set to true. Therefore, `true && false == false`.
9. The variables pop and nm are instance variables and there is only one constructor (public School(String a, int b){}). Two instance variables + one constructor = 3.
10. The two methods are rename(String s) and getPopulation().
11. A memory address would be output because no toString() method is implemented.
12. 5>> 2+3&3 = 5 >>5&3 = 0&3 = 0
13. b=20, a=40, Math.min(20,40) = 20
14. Starting at position 3, the string "biochemsics" starts with "chem."
15. Both C and D work. In Java 7, the diamond operator allows D to be legal
16. The 0th position is removed, and thus 62 + 7 = 69 is output. The + operator turns numList.get(2) into an integer.
17. 077 = 77_8 and the "%d" turns this value into base 10. Thus, $77_8 = 63_{10}$
18. First, u+r = 5.23+8.2 = 13.43. t%(u+r) = 7 % 13.43 = 7.0. h+7.0 = 9 + 7.0 = 16.0. This is cast to an int and then into a float, the resultant answer is 16.0.
19. The difference in lengths between the two strings is the result because there is no difference between the two strings. Therefore, 4-11 = -7
20. <*1> must be filled in with "Play" otherwise an error for incompatible types will come up due to trying to add instances of Play to the array.
21. The class is sorted based on the average value of all of the elements in the string that is passed in during instantiation. The array of Play's is sorted in descending order based on the average value of all the elements in the string. Thus in this case, "hue" = (104+117+101)/3=107, "aa" = (97+97)/2 = 97, "za" = (122+97)/2 = 109. Thus in decreasing order it is "za", "hue", "aa". The first character of each element is outputted resulting in "zha".
22. The output will always be true because list2 simply points at list. Once list is modified, list2 is also modified. Therefore, the output is always true and the randoms do not matter.
23. Method ez requires a string that has odd length otherwise a runtime error will occur due to the base case condition (i.e. A string of length 2 will be cut to a string of length 0 and the base case will not catch it resulting in a StringIndexOutOfBoundsException. Otherwise, if the string is even length, then the method works fine which concatenates every character of odd position (starting with index 1).
24. See explanation in previous question, ez requires a string of odd length or a StringIndexOutOfBoundsException will occur which is a runtime error.
25. Math.log1p(double x) returns the sum of the argument and 1, the method also returns a double. Math.Ln(e-1+1)=1. Therefore the output is 1.0.
26. The PriorityQueue sorts the elements in ascending order when the elements are individually removed from the PriorityQueue. This is because the PriorityQueue implements a min heap data structure. Therefore, the element that is pulled out is always the smallest element in the set, resulting in a list that is sorted in ascending order. Thus, the answer is B.

27. When outputting the PriorityQueue, it is simply a level-order traversal of the min heap. Thus, if you add the list {6, 2, 10, 9, 4, 15} into a PriorityQueue you will get a min heap that when level-order traversed outputs to be [2, 4, 10, 9, 6, 15]. Thus, the answer is A.
28. Because the PriorityQueue implements a min heap, the best answer is heap sort.
29. The algorithm is a classic example of the coin problem. The basis of the algorithm is dynamic programming and merely calculates the smallest number of coins necessary to create the sum that is input given a pool of coin values to choose from. Therefore, the smallest number of coins in order to create 6 from a set of coins with values {2, 3, 6} is 1.
30. See above explanation. The smallest number of coins needed to create a value of 212 is 11 (8 x 25 + 1 x 10 + 2 x 1).
31. HashMap, TreeMap, and LinkedHashMap are all valid to fill in <*1>. The difference between the types of Maps is based on the iteration order when going through the keys. HashMap guarantees no order, the keys in TreeMap are sorted according to the compareTo() method, and LinkedHashMap iterates through the keys based on the order which the entries were put in the map.
32. The algorithm is a modified frequency counter. It functions normally to count frequency of different characters, but double counts all uppercase letters (or characters with an ASCII value less than 94 to be exact). Therefore, since there are 3 capital O's, the output should be 6.
33. In order to solve this problem, one must understand the concept of complement 2's. The binary representation of -2 can be found by taking the binary representation of 2, flipping all the bits, and adding 1 (00000000000000000000000000000010 -> 11111111111111111111111111111101 -> 11111111111111111111111111111110). The bits are shifted right by 2 and because it is the >>> operator, the sign bit is shifted also resulting in 00111111111111111111111111111111 which is equal to 1073741823 in base 10. Finally the result is shifted back left by 2 bits resulting in 11111111111111111111111111111100 which represent -4 in complement 2's.
34. Simply take length1/length2 = 2,496/649 = ¼. (1/4)^5 = 1/1024. 1/1024*2048 = 2 seconds.
35. Turns out it actually isn't a trap. This is a legitimate program.
36. G=P-[L-(B+I)]^A/Y^S
    G=P-[L-(BI+)]^A/Y^S
    G=P-[LBI+-]^A/Y^S
    G=P-[LBI+-]A^/YS^
    G=P-[LBI+-]A^YS^/
    G=PLBI+-A^YS^/-
    GPLBI+-A^YS^/-=
37. G=P-[L-(B+I)]^A/Y^S
    G=P-[-L+BI]^A/Y^S
    G=P-^[-L+BI]A/^YS
    G=P-/^[-L+BI]A^YS
    G=-P/^-L+BIA^YS
    =G-P/^-L+BIA^YS
38. The electronic diagram can be rewritten as [(A and B) nor (C and D)] and E. The following values of A, B, C, D, and E result in a true output. Therefore, there are 9 combinations of A, B, C, D, and E that result in an output of true.
    A = false B = false C = false D = false E = true
    A = false B = false C = false D = true E = true
    A = false B = false C = true D = false E = true
    A = false B = true C = false D = false E = true
    A = false B = true C = false D = true E = true
    A = false B = true C = true D = false E = true
    A = true B = false C = false D = false E = true
    A = true B = false C = false D = true E = true
    A = true B = false C = true D = false E = true
    The fast way to solve this problem is to realize that there are 3 combinations of A and B (T and F, F and T, F and F) that result in false. Thus, because C and D is the same, just multiply the 3 by 3 which results in 9.

39. $(X * (\overline{Y} + \overline{Z})) + (Z * Y * X)$
$(X * (\overline{Y} + \overline{Z})) + (X * Y * Z)$
$(X * \overline{(Y * Z)}) + (X * Y * Z)$
$(X * (\overline{(Y * Z)} + (Y * Z)))$
$(X * 1)$
X

40. The reverse in-order of a tree is simply the reverse of an in-order traversal (as the name implies). In the in-order traversal the nodes are traversed in an LDR (left, data, right) fashion. In a reverse in-order traversal the nodes are traversed in an RDL (right, data left) fashion. Thus, the in-order traversal yields BIGPLAYS which is reversed to yield the answer SYALPGIB.