★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 REGION

Questions (+6 points for each correct answer, -2 points for each incorrect answer)

1) E	11) A	21) B	31) B
2) C	12) B	22) E	32) A
3) A	13) D	23) D	33) D
4) E	14) A	24) A	34) C
5) D	15) C	25) E	35) E
6) E	16) A	26) C	36) B
7) A	17) D	27) D	37) C
8) B	18) C	28) B	38) A
9) E	19) A	29) B	*39) (A*C) + (B*D)
10) C	20) E	30) E	40) 58

KEY

** See "Explanation" section below for alternate, acceptable answers.*

Note: Correct responses are based on **Java SE Development Kit 8 (JDK 8)** from Sun Microsystems, Inc. All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g., "error" is an answer choice) and any necessary Java SE 8 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used.

Explanation

1) E $3D_{16}$ * 13_8 = 1010011111_2 = 22133_4 = 1237_8 = 671_{10} = $29F_{16}$

2) C `(0.4 + 1.2 * 8) / 2 = (0.4 + 9.6) / 2 = 10.0 / 2 = 5.0`

3) A `"%(08d"`: Negative values are to be formatted with parentheses instead of a negative sign and the parentheses count toward the 8-character field width. Number should be zero-padded to fill out the remaining field width.

4) E The `replace()` method creates a new `String` in which each substring that matches the literal target sequence (1st parameter) is replaced with the specified literal replacement sequence (2nd parameter). The original `String` is not modified. Hence, `mixed` is never actually modified in the code segment, `zero` is a reference to the unmodified `mixed`, and `ones` references a modification of `mixed`.

5) D `!(w <= x && y != z) = !(w <= x) || !(y != z)`, by DeMorgan's Law.

6) E `raw = -2.5; floor = 3.0; ceil = 2.0; Math.pow(3.0, 2.0) = 9.0`

7) A `hund = 236; ten = 23; one = 6; 23 + (10 * 6) = 83`

8) B `(12345 / 9) = 1371 > 1000; (12345 * 4) = 49380 < 50000`

9) E Variable `digits` overflows when its value exceeds `127` (i.e., `Byte.MAX_VALUE`). `128` overflows to become `-128`. `-256` underflows to become `0`. The resulting values for the variable `digits` at the start of each iteration are `1, 2, 4, 8, 16, 32, 64, -128, 0, 0, 0, 0, ...` causing the output to be `"1248624-80000..."` (Infinitely repeating zeroes).

Copyright © 2016 University Interscholastic League

★ANSWER KEY – CONFIDENTIAL★

10) C For each index position from `i = 6` through `i = 1`, compares values to either side of `i` and records in `w` the rightmost value if it is less than the leftmost of the pair being compared. Values for `w` include `–1, 5, 2, 3`. Method returns the final value of `w = 3`.

11) A The `Scanner` is initialized with a `String` literal (`"seuss.txt"`) as the source rather than an actual `File` object. Parsing through the 9 characters in the `String`, `"seuss.txt"`, results in only a single line of text being read and added to the `List`. In order for the actual file contents to be parsed, the `Scanner` would have needed to be initialized as `new Scanner(new File("data.txt"))`.

12) B The contents of the `eight` array include `[0.0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875]`.

13) D `((me – ((you / us) * me)) + (you * me)) = ((5 – ((24 / 3) * 5)) + (24 * 5))`

14) A `public class Random extends Object implements Serializable { ... }`

15) C `all = [`*reference to some*`]; some = [`*reference to all*`, 0, 2]`

16) A Default value for `boolean` primitive is `false`. Default value for `Boolean` object is `null`.

17) D `max = 2147483647; min = –2147483648`. Due to overflow, `–min = –2147483648` as well.

18) C Answer choice A produces random integers from the set of `[0, 1, 2]`. Answer choices B and D produce random integers from the set of `[–1, 0]`.

19) A `morse = ".-...---...----."`. String is split using the regular expression, `"."`, which splits on *any* single character (`"."` = regex wildcard pattern character) and would produce an array of 15 empty strings, However, `split()` does not include trailing empty strings in the resulting array, leaving `dashes` as an empty array.

20) E Sequential search finds index of last occurrence of `item` in `data`.

21) B Strings are removed from the `PriorityQueue` in ascending, lexicographic order. `'2' = 50, 'F' = 70, 'T' = 84, 'f' = 102`.

22) E After bitshifting 31 by 2 places to the left, `scan` $= 01111100_2 = 124_{10}$.

23) D $52_{15} = 77_{10}$

24) A Sums up the total number of positions each items are shifted during each insertion. The `'E'` is shifted by 0 positions. `'G'` is shifted by 1 position. `'I'` is shifted by 2 positions. `'O'` and `'N'` are each shifted by 3 positions. `0 + 1 + 2 + 3 + 3 = 9`.

25) E The resulting array is sorted in descending Unicode order.

26) C Removes and inserts each item in the list into descending order.

27) D Insertion sort yields $O(N^2)$ performance in the average and worst cases

28) B The regular expression requires the string to end with a non-digit character (e.g., `[^0–9]`).

29) B Returns a post-order traversal of the heap rooted at index `n`, where index [1] of `String src` represents the root of the overall heap.

30) E Throws a `StringIndexOutOfBoundsException` when `n = src.length()`.

31) B Any value XOR'ed with its bitwise complement will result in a value containing all 1 bits (i.e., 2's complement representation of –1).

32) A `base` references the head (node `[1]`) of a bi-directional, linked list consisting of `Disc` objects numbered as follows:
{WEST} `null ← [1] ↔ [7] ↔ [6] ↔ [5] ↔ [4] ↔ [3] ↔ [2] → null` *{EAST}*

33) D `base` references the head (node `[1]`) of a bi-directional, linked list consisting of `Disc` objects numbered as follows:
{WEST} `null ← [1] ↔ [7] ↔ [6] ↔ [3] ↔ [2] → null` *{EAST}*

34) C Since the initial `state` of node `[3]` is `false`, its `state` and the `state` of node `[6]` (to the `west` of node `[3]`) are each inverted to `true` (producing an output of "H" for each of those 2 nodes).

35) E Traversing `east` from node `[4]` leads to node `[3]` and node `[2]`, but traversing `west` from node `[2]` leads to node `[3]`, node `[6]`, node `[7]`, and node `[1]`.

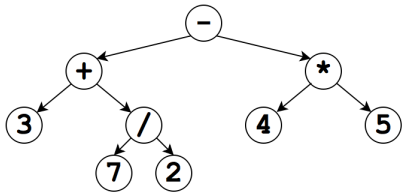
36) B Each `Disc` object is a node in a bi-directional, linked list of `Disc` objects.

37) C Paths: `AGD`, `ABCGD`, `AFECGD`

KEY

★ANSWER KEY – CONFIDENTIAL★

38) A RPN is the post-order traversal of the following expression tree:



39) Any answer that equivalently expresses "(A Logical-AND C) Logical-OR (B Logical-AND D)" is acceptable (use of parentheses for correctly enforcing order of operations is *optional*):

```
(AC) + (BD)
(A * C) + (B * D)
(A && C) || (B && D)
(A and C) or (B and D)

(CA) + (BD)
(C * A) + (B * D)
(C && A) || (B && D)
(C and A) or (B and D)

(AC) + (DB)
(A * C) + (D * B)
(A && C) || (D && B)
(A and C) or (D and B)

(CA) + (DB)
(C * A) + (D * B)
(C && A) || (D && B)
(C and A) or (D and B)
```

```
(BD) + (AC)
(B * D) + (A * C)
(B && D) || (A && C)
(B and D) or (A and C)

(DB) + (AC)
(D * B) + (A * C)
(D && B) || (A && C)
(D and B) or (A and C)

(BD) + (CA)
(B * D) + (C * A)
(B && D) || (C && A)
(B and D) or (C and A)

(DB) + (CA)
(D * B) + (C * A)
(D && B) || (C && A)
(D and B) or (C and A)
```

40) $11011101_2 = -35$; $01011101_2 = 93$; $-35_{10} + 93_{10} = 58_{10}$

KEY