★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 DISTRICT 1

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

1) E	11) E	21) B	31) A
2) D	12) C	22) D	32) B
3) B	13) D	23) A	33) E
4) C	14) C	24) B	34) D
5) A	15) D	25) C	35) C
6) E	16) B	26) E	36) C
7) B	17) A	27) D	37) D
8) C	18) D	28) A	38) B
9) D	19) B	29) A	*39) B * (A + C)
10) B	20) E	30) B	*40) 734*–

** 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 $100101_2 + 1111_2 = 110100_2 = 64_8 = 52_{10} = 34_{16} = 1G_{36}$

2) D `(25 + 8) / 8 = 33 / 8 = 4` (integer division)

3) B Loop iterates backward through array, printing Unicode characters in reverse order all on 1 line.

4) C `substring(int begin, int end)`: Returns the substring from index `begin` through index `(end - 1)`.

5) A

P	Q	R	**X**	A)	B)	C)	D)	E)
0	0	0	**0**	0	1	0	0	0
0	0	1	**0**	0	1	0	1	0
0	1	0	**1**	1	1	1	1	0
0	1	1	**1**	1	1	0	1	0
1	0	0	**0**	0	0	1	0	0
1	0	1	**0**	0	1	0	1	0
1	1	0	**0**	0	1	1	0	0
1	1	1	**1**	1	1	0	1	1

6) E The code segment can produce outputs in the range of 16 through 35, inclusive.

7) B `huey = -10; dewey = -30; louie = -33; -33 % -10 = -3`

★ANSWER KEY – CONFIDENTIAL★

8) C The `switch` matches on the `case` where `test = 5`, prints "`five`", increments `test` to 6, then breaks out of the `switch-case` statement before printing the final value of `test` (6).

9) D Prints an "`@`" when `at = 3`, 9, 27, 81, 243, 729, and 2187. Exits the loop when `at = 6561`.

10) B `9 + 3 + 5 = 17`

11) E Value of `alfa`, `bravo`, and output at the point of the `print()` invocation in each iteration of the loop:

```
 alfa: 15 13 11  8  7  5
bravo: 28 11 30 15 33 45
Output: 13 11  8  7  5  0
```

12) C `half = 0 + 1000 + 500 + 250 + 125 + 62 + 31 + 15 + 7 + 3 + 1 = 1994`

13) D
```
= (23 | ((6 << 3) ^ (7 & 13)))
= (23 | (48 ^ (7 & 13))
= (23 | (48 ^ 5))
= (23 | 53)
= 55
```

14) C The barcode (UPC) serves as the lookup *key* and the product price is the *value* associated with that key. O(1) to add/update product prices. O(1) to look up the price of a product.

15) D
```
bytes = []
bytes = [0]          (when i = 0, set bytes[0/2] = 0/3)
bytes = [0, 6]
bytes = [0, 6, 12]
bytes = [0, 1, 12, 18]          (when i = 3, set bytes[3/2] = 3/3)
bytes = [0, 1, 12, 18, 24]
bytes = [0, 1, 12, 18, 24, 30]
bytes = [0, 1, 12, 2, 24, 30, 36]          (when i = 6, set bytes[6/2] = 6/3)
bytes = [0, 1, 12, 2, 24, 30, 36, 42]
bytes = [0, 1, 12, 2, 24, 30, 36, 42, 48]
bytes = [0, 1, 12, 2, 3, 30, 36, 42, 48, 54]          (when i = 9, set bytes[9/2] = 9/3)
```

16) B Return value, `i`, increments to index of first occurrence of `x` (`i = 4`). No exceptions are thrown and the `finally` clause is always executed. The method prints "`3`" before returning and the client class prints "`4`".

17) A Return value, `i`, increments beyond the end of the array causing an `ArrayIndexOutOfBoundsException` to be thrown when `i = 10`. The exception is caught by the first `catch()` clause (`code = 1`) and the `finally` clause is always executed (`code = 13`). Note that `ArrayIndexOutOfBoundsException extends IndexOutOfBoundsException extends RuntimeException extends Exception`. The method prints "`13`" before returning and the client class prints "`10`".

18) D "`That`" < "`This`" < "`the Other`" when compared lexicographically (case-sensitive).

19) B $19_{10} = 14_{15}$

20) E The regular expression requires 2 `b`'s following 1 or more leading `a`'s.

21) B Recursively produces a pre-order traversal of the tree whose in-order traversal is the parameter `String s`.

22) D Outer loop iterates `six` through values of 4, 6, and 8. Inner loop iterates `two` through values of 2 through 3, 3 through 5, and 4 through 7, respectively for each pass through the outer loop.

23) A Sorts the array into descending order using selection sort.

24) B Selection sort yields $O(N^2)$ performance in the best, average, and worst cases.

25) C Selection sort performs `N` swaps, incrementing the return value from `-1` through `N-1` (i.e., 7), including cases where an item is swapped with itself (i.e., `i == m`).

26) E `data` is a `char[]` sorted in descending order by Unicode value.

27) D `chunks = ["t", "", "", "o", "", "e"]`. The regular expression matches on "`o b`", "`e o`", "`r n`", "`t t`", and "`o b`".

28) A `remove()` causes an item to be removed from the queue, but `peek()` does not remove the item from the queue.

★ANSWER KEY – CONFIDENTIAL★

29) A `((2 + 3) ^ (4 + 1)) = 5 XOR 5 = 0`

30) B `grid = [[1, 5, 8, 10], [5, 6, 9], [8, 9], [10]]`

31) A `amounts = 5.00 + 2.00 + 10.50 = 17.50; tips = 0.00 + 0.00 + 0.00 = 0.00`

32) B The `pay()` method defined in the `Cheapo` class uses the `private rate` field declared in the `Cheapo` class (`0.00`) and does not use either the `private rate` field declared in the `MoneyBags` subclass (`1.50`) or the `private rate` field declared in the `Customer` interface (`1.20`). But the `getRate()` method in the `MoneyBags` subclass uses the `private rate` field declared in the `MoneyBags` subclass (`1.50`).

33) E `Customer` is an `interface`. It cannot be directly instantiated with `new Customer()`.

34) D

Keys	Values
one	~~two~~ four
two	two
four	five
five	one

35) C Pre-order: DERCSANLMBU. In-Order: CRSEANDMLUB. Level-by-level: DELRAMBCSNU

36) C `X = (Q * R) + (P * R) + (P * Q)`. If any 2 inputs are 1 (i.e., Q and R, P and R, or P and Q), the output is 1. Note that the correct answer choice only addresses cases in which "exactly 2" inputs are 1 and makes no statement about the output if all 3 inputs are true (i.e., the output could be either 0 or 1 in that case).

37) D $100_{10} = 01100100_2$; $-100_{10} = 10011100_2$; 1's complement of $-100_{10} = 10011011_2$

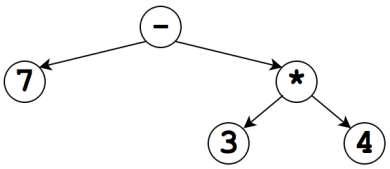
38) B

P	Q	R	**X**	**Y**
0	0	0	**1**	**0**
0	0	1	**0**	**0**
0	1	0	**1**	**1**
0	1	1	**1**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**0**
1	1	1	**0**	**1**

39) Any answer that equivalently expresses "(NOT B) Logical-AND (A Logical-OR C)" is acceptable (use of parentheses for correctly enforcing order of operations is required):

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

40)



Postfix (reverse Polish) notation: `734*–`
Prefix (Polish) notaton: `–7*34`
Infix notation: `7–(3*4)`