★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE WRITTEN TEST – 2016 DISTRICT 2

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

1) B

2) A

3) D

4) A

5) D

6) A

7) B

8) D

9) A

10) E

11) B

12) B

13) E

14) B

15) C

16) B

17) D

18) C

19) A

20) C

21) D

22) B

23) C

24) C

25) D

26) C

27) E

28) B

29) D

30) B

31) C

32) A

33) E

34) E

35) A

36) C

37) A

38) B

39) /+283

*40) (X * Y) + Z

** 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) B $110011_2 + 10111_2 = 1001010_2 = 2202_3 = 112_8 = 74_{10} = 4A_{16}$

2) A `20 % (20-15) = 20 % 5 = 0`

3) D The loop iterates through the integer equivalents of the characters in the array, but prints each as a Unicode character, 1 character per line.

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

5) D

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

6) A The code segment can produce outputs in the range of 7 through 17, inclusive.

7) B `alfa = 25;` `bravo = -75;` `charlie = -64;` `-64 % 25 = -14`

★ANSWER KEY – CONFIDENTIAL★

8) D The switch matches on the case where test = 4, prints "four", increments test to 7, prints "eight", then exits the switch-case statement before printing the final value of test (7).

9) A Prints a "#" when pound = 4, 16, and 256. Exits the loop when pound = 65536.

10) E
```
b = 107 + 105 + 99 + 107 + 101 + 108 = 627
c = 107 + 109 = 216
(b + c) / 2 = (627 + 216) / 2 = 843 / 2 = 421 (integer division)
```

11) B Value of cat, dog, and output at the point of the print() invocation in each iteration of the loop:

dog:	1	11	9	1	6
cat:	11	9	1	6	3
Output:	**-10**	**2**	**8**	**-5**	**3**

12) B sum = 15 + 30 + 60 + 120 + 240 + 480 = 945

13) E
```
= ((28 ^ (77 >> 3)) | (15 & 27))
= ((28 ^ 9) | (15 & 27))
= ((28 ^ 9) | 11)
= (21 | 11)
= 31
```

14) B O(1) to add new customers to the tail of the queue. O(1) to remove customers from the head of the queue.

15) C
```
longs = []
longs = [0]                  (when i = 0, set bytes[0/3] = 0*2)
longs = [0, 4]
longs = [4, 4, 8]            (when i = 2, set bytes[2/3] = 2*2)
longs = [4, 4, 8, 12]
longs = [4, 8, 8, 12, 16]            (when i = 4, set bytes[4/3] = 4*2)
longs = [4, 8, 8, 12, 16, 20]
longs = [4, 8, 12, 12, 16, 20, 24]           (when i = 6, set bytes[6/3] = 6*2)
longs = [4, 8, 12, 12, 16, 20, 24, 28]
longs = [4, 8, 16, 12, 16, 20, 24, 28, 32]           (when i = 8, set bytes[8/3] =8*2)
longs = [4, 8, 16, 12, 16, 20, 24, 28, 32, 36]
```

16) B The second 1 in the string, 01**1**001011, breaks the pattern specified by the regular expression, 0*1(**0+**1+)*.

17) D pop() causes an item to be removed from the stack, but peek() does not remove the item from the stack.

18) C Recursively produces a post-order traversal of the tree whose in-order traversal is the parameter String s.

19) A Recursively finds the index of parameter, b, in array, a, using binary search.

20) C Binary search yields O($\log_2$ N) performance in the average and worst cases.

21) D Finds the index of 'e' in data. Note that data is not sorted, so the method actually happens upon the 2nd occurrence of 'e' in the array.

22) B The help() method returns a sub-array containing the elements of array a from index positions x through y - 1, inclusive.

23) C $36_{10} = 2a_{13}$

24) C slices = ["ru", "umpers"]. The ".*" in the regular expression is greedy and matches on all characters between the first and last 'b' in the string ("bber baby buggy b").

25) D "2 fish" < "Blue fish" < "Red fish" < "one fish" when compared lexicographically (case-sensitive).

26) C Results in an ArithmeticException ("/ by zero") when attempting to divide a[7] = 5 by a[8] = 0. The exception is caught by the first catch() clause (code = 1) and the finally clause is always executed (code = 13). Note that ArithmeticException extends RuntimeException extends Exception.

27) E No exceptions are thrown and the finally clause is always executed (code = 3).

28) B Outer loop iterates one through values of 4, 6, and 8. Inner loop iterates ten through values of 2, 2 through 3, and 2 through 4, respectively for each pass through the outer loop. Note that the output concatenates the values of ten and one.

★ANSWER KEY – CONFIDENTIAL★

29) D $X = (P * R) + (\overline{P} * Q * \overline{R})$. When the values of P and R are different from each other, the output is 0.

Note that the correct answer choice only addresses cases in which P != R and makes no statement about the output if P == R (i.e., the output could be either 0 or 1 in that case).

30) B `((2 ^ 3) | (6 - 4)) = ((2 XOR 3) OR 2) = (1 XOR 2) = 3`

31) C `wages = 50.0 + 25.0 = 75.0`. The `paycheck()` and `comment()` methods defined in the `Worker` class bind with the `bonus` field declared in the `Worker` class (`1.25`) and never bind with either the `bonus` field declared in the `Slacker` subclass (`0.75`) or the overridden `rate` field declared in the `Employee` interface (`1.00`). But the `toString()` method in the `Slacker` subclass binds with and references the `bonus` field declared in the `Slacker` subclass (`0.75`).

32) A `wages = 50.0 + 25.0 = 75.0`

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

34) E

Keys	Values
Dopey	Doc
Grumpy	~~Bashful~~ Dopey
Happy	Doc
Sleepy	Sneezy

35) A `grid = [[1, 2, 3, 4], [2, 6, 7], [3, 7], [4]]`

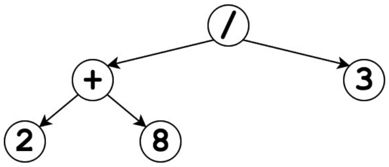
36) C

P	Q	R	X = !(R && P) && (!P && Q)	Y = (!P && Q) && (!Q || R)
0	0	0	**0**	**0**
0	0	1	**0**	**0**
0	1	0	**1**	**0**
0	1	1	**1**	**1**
1	0	0	**0**	**0**
1	0	1	**0**	**0**
1	1	0	**0**	**0**
1	1	1	**0**	**0**

37) A Post-order: `OIMENADSRLP`. In-Order: `OIAMNEPDLSR`. Level-by-level: `PALINDROMES`

38) B $111_{10} = 01101111_2$; $-100_{10} = 10010001_2$; 1's complement of $-100_{10} = 10010000_2$

39)



Postfix (reverse Polish) notation: `28+3/`
Prefix (Polish) notaton: `/+283`
Infix notation: `(2+8)/3`

40) Any answer that equivalently expresses "(X Logical-AND Y) Logical-OR Z" is acceptable (use of parentheses is optional):

```
XY + Z             YX + Z             Z + XY             Z + YX
(X * Y) + Z        (Y * X) + Z        Z + (X * Y)        Z + (Y * X)
(X && Y) || Z      (Y && X) || Z      Z || (X && Y)      Z || (Y && X)
(X and Y) or Z     (Y and X) or Z     Z or (X and Y)     Z or (Y and X)
```