★ANSWER KEY – CONFIDENTIAL★

UIL COMPUTER SCIENCE – 2025-2026 INVITATIONAL B

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

*1) 861	11) C	21) B	31) F
2) E	12) A	22) D	32) B
3) B	13) C	23) C	33) C
4) D	14) B	24) H	34) F
5) A	15) E	25) F	35) C
6) C	16) D	26) B	36) B
7) D	17) A	27) C	37) D
8) A	18) C	28) C	38) C
9) E	19) E	29) F	*39) 10111101
10) B	20) D	30) A	*40) 19

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

Note: Correct responses are based on **Java SE Development Kit 22 (JDK 22)** 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 22 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used.

Explanations:

1.	861	Convert 345 from base 10 to base 5 → 2340_5 Now treat that number as base 7 → $2340_7 = \boxed{861_{10}}$
2.	E	After doing the pre/post increment/decrement the equation look like… `5 + 2.5 * 3 / (0) + 3 - (-2.5);` Double division by zero results in `Infinity`.
3.	B	The code prints a quote, then `10/4` as `2`, then the literal characters `\n`, then `10/4.0` as `2.5`, then two backslashes from `\\\\`, then a final quote from `\"`, giving `"2\n2.5\\"`
4.	D	https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#intern--
5.	A	Boolean evaluation with order of precedence.
6.	C	`Math.abs(Integer.MIN_VALUE)` overflows and stays `-2147483648`, while `Math.floorMod(m, 7)` returns the nonnegative remainder `5` (unlike `m % 7`, which would be `-2`).
7.	D	It becomes `4*6/8=3.0`, then `+2`, and subtracting (`6-9=-3`) adds `3`, giving `8.0`.
8.	A	It enters case 1 with `x=2`, increments to `3` so adds `A`, falls through to case 2 where `x++==3` adds `C` and leaves `x=4`, printing `AC4`.
9.	E	It prints `02` on the first pass (`i` goes 0→1 and `j` goes 3→2) and `11` on the second (`i` goes 1→2 and `j` goes 2→1), then stops because `i < j` is `false` and never prints `|`, so the output is `0211`.
10.	B	`a[i]++` uses `a[0]=2` as the target index (then makes `a[0]=3`), the RHS becomes `a[1]+a[1]=0`, and since `b` aliases `a` it assigns `a[2]=0`, leaving `3 0 0 0 2` → `30002`.
11.	C	`useRadix(8)` makes `nextInt()` read `7` and `010` as octal (`7` and `8`) while `nextInt(10)` reads `8` and `09` as decimal (`8` and `9`), and `getParent()`/`getName()` produce `dir` and `nums.txt`, so it prints `dir|nums.txt|32`.
12.	A	Processing digits 8, 0, 5, the parity check happens *before* the `d==0` correction, so `prod` becomes 1→9→9→54 and `sum` ends 0→8→7→12, printing `12|54`.
13.	C	`x + y << z` is `(x+y)<<z = 3<<3 = 24`, `x | y & z` is `x|(y&z) = 1|(2&3)=3`, and the condition is `true` (due to `||` / `&&` precedence and short-circuiting) so it adds 4, giving `24 ^ (3+4) = 24 ^ 7 = 31`.
14.	B	In two's complement, bitwise NOT is equivalent to negating and subtracting one, so `~x` always equals `-x - 1`.
15.	E	`v.remove(1)` removes index `1` of the `subList` (printing `2`), then modifying `a` directly invalidates `v`'s expected modification count so the `v.size()` call throws a `ConcurrentModificationException`.
16.	D	`StringIndexOutOfBoundsException` and `ArrayIndexOutOfBoundsException` are different, so the exception is not caught. The `finally` block will always be executed, so `3` is printed before the error is output.
17.	A	`poll()` is only for objects that are children of `Queue`, and there is no `remove()` method for `Stack` with no arguments, so only `pop()` will work.
18.	C	Stacks are FILO designed, so both `add()` and `pop()` will interact with the top of the stack.
19.	E	Insertion and Deletion are $\mathcal{O}(1)$ operations for stacks, and the loop is constant time $\mathcal{O}(12) = \mathcal{O}(1)$. So the total is $\mathcal{O}(1) \cdot \mathcal{O}(1) + \mathcal{O}(1) = \mathcal{O}(2) = \mathcal{O}(1)$, so it is constant time.
20.	D	Lines 1 and 2 are totally valid, no errors. Line 3 causes a runtime error, but line 4 causes a compile time error, which will happen before the code is run to trigger line 3's error.
21.	B	Recursive Tracing.
22.	D	Recursive Tracing.
23.	C	You implement an `interface` and extend a `class`, but you must `extend` first when defining the child
24.	H	All 3 methods from the `interface` and `abstract class` must be implemented by the child. Additionally, you must define a `super()` call with one string argument in the `iPhone` constructor, as there is no default constructor in the `abstract class`, so the constructor for `iPhone` must be defined.

25.	F	You cannot decrease the scope of a method from a parent `class`/`interface`, and the methods in an `abstract class` are of scope smaller than anything except `private` by default if not defined. Can choose any scope except `private`.
26.	B	You cannot decrease the scope of a method from a parent `class`/`interface`, and the methods in an `interface` are of bigger scope than anything except `public` by default if not defined. Can only choose `public` scope.
27.	C	The output will be the phone number `8328328321` followed by a space, followed by the total number of apps, which is `5`.
28.	C	The regex pattern is equivalent to a capital letter followed by any number (even 0) of lowercase letters, or any number of non-digit characters (`\\W` also excludes `\\D` entirely, so we take the less exclusive since it is an `|`).
29.	F	All 4 instantiations are legal.
30.	A	Selection sort is $O(N^2)$, so with 4,000 items, we square it, $16{,}000{,}000 = 16$ seconds, $1{,}000{,}000 = 1$ second. For 2,000 items → $4{,}000{,}000 = 4$ seconds.
31.	F	The code will run without error as is. However, answer choices B through E will also cause the code to run without error (albeit, producing different output). Thus, answer choices A through E are all accurate, but F is the most accurate, per the question's request.
32.	B	Since `L2` is not contained within a static block, it will not be executed ever. However, `L3` is contained within a static method that is called when running the program, and thus `L3` is the only line that gets printed. For integers with underscores in the format, the underscores are omitted when printed to the console.
33.	C	Despite the main method being automatically called, all static blocks are executed before that, and so `L2` is printed first, and then `L3`.
34.	F	Post-increment/decrement operations have higher precedence than pre-increment/decrement. Method reference has the highest precedence, and lambda has the loweest.
35.	C	The main tricky thing with this question is the "<" index argument. This specifies that the argument from the previous placeholder should be used. Note that when using the "<", this does not consume what would have been the next-used argument fed to the `printf` method. There is also no issue in providing more arguments to `printf` than what actually get used. So the order of arguments that get used are `7 + 3`, then `(int) Math.pow(2,3)`, then `(int) Math.pow(2,3)` again, then `3.14159`, then `3.14159` again, and lastly `Math.PI`. `%n` prints platform-independent newline character.
36.	B	First, Alice is correct, not Bob. This leaves answer choices A and B. The fact that $n-1$ edges are formed in a graph with n vertices does not guarantee that a tree is formed – it could form a forest instead. The goal with the proof here is to mathematically show that the specific construction method of those $n-1$ edges guarantees that a tree is formed. Option A is effectively circular logic, and just re-states what the proof goal is as an assumption despite having not shown that yet. Option B on the other hand accurately explains why this specific construction pattern guarantees that a tree is formed, and thus is the most accurate.
37.	D	Option A is false as stated above. Option C is false – the only other way for $n-1$ unique edges of an n vertex graph to not form a tree is if it has multiple connected components. However, since there are $n-1$ unique edges, it can be shown that if there are multiple connected components, then at least one edge from one of the multiple connected components must form a cycle and thus showing that a cycle is impossible to form is both sufficient and necessary to show the accuracy of this algorithm. Option E is also false. Often generalizing a proof increases its strength rather than loosens it, as we make fewer assumptions regarding the actual values of the graph (and thus is harder to prove). The assumption regarding form that is made is perfectly reasonable since a cycle must exist per the proof by contradiction's assumption. Option D is where the main issue exists. Consider vertex v in the cycle. This vertex appears in two different edges that are a part of the cycle we are assuming exists (i.e., (u,v) and (v,w)). Since we "assumed without loss of generality" the fact that the cycle was of the form $u \rightarrow v \rightarrow w \rightarrow \cdots \rightarrow u$ doesn't mean that the second edge coming out of u is necessarily the…

		… problematic one – in another situation it might have instead been v. Moreover, What the algorithm guarantees is only that after an edge (u, v) is added, either u or v is removed from the candidate set, but it does not guarantee that u specifically is removed. Therefore, u may remain in V and be selected again in later iterations, allowing it to participate in multiple edges. The main way this would happen is if there are other vertices that are not a part of the cycle, but instead connected to nodes that are a part of the cycle (e.g., vertex u). So there may be more than 2 edges coming into/out of u, while u is still a part of some cycle (per the assumption).
38.	C	The initial state is q_1 and the two accepting states are q_1 and q_2. Only strings which end in an accepting state are included in the language of a DFA; thus, anything that visits q_3 cannot be generated by the DFA (as q_3 is a trap state since you cannot leave once entering). Here $*$ means "0 or more times," $+$ means "or", and when two symbols appear directly next to one another, it means "and" (much the same way that Java regex works). Thus, the only regular expression which guarantees that the final state is either q_1 or q_2, and abides by the state transition functions, is option C: `((0(0+1))*)+((0(0+1))*0)` This regular expression reads as: "Either (i) 0 followed by either 0 or 1, all zero or more times, or (ii) 0 followed by either a 0 or 1, all zero or more times, followed by a 0."
39.	10111101	67_{10} in binary $= 01000011_2$, flip bit $= 10111100_2$, add $1_2 = 10111101_2$
40.	19	Just write out all 32 combos, there's some simplification, but in this case it is easier to just treat the expression as is, and plug all combos in and try.