**Template numbers and their corresponding concepts**

**Arithmetic Expressions**

* 300 – Associativity of subtraction (3 + 4 – 5)
* 203 – Precedence of division (6 + 7 / 3)
* 305 – Associativity of Multiplication (2 \* 3 \* 1.5)
* 424 – Precedence of Remainder (9 % 5 – 5 % 9)
* 209 – Precedence of Multiplication (8 – 2 \* 5)
* 310 – Associativity of Division (9 / 3/ / 2.0)
* 433 – Correct evaluation of Multiplication (5 / 3 + 3 \* 5)
* 428 – Precedence of Addition (3 / 6 + 3 % 6)
* 420 – Precedence of Subtraction (8 / 5.0 – 8 / 5)
* 413 – Evaluation of Remainder on Real Operands (12 % 5 + 5 % 12.0)
* 153 – Associativity of Addition (4.3 + 5 + 3)
* 126 – Correct evaluation of Subtraction (8 - -5)
* 322 – Dividing by Zero (9 / 3 / 0)
* 441 – Remainder when Divided by Zero (8 – 6 – 4 % 0)
* 622 – Precedence of Parenthesis (12 / 3 \* (7 – 5))
* 654 – Associativity of Parenthesis (14 / ((6 – 3) + 2))

**Relational Expressions**

* 231 – Correct evaluation of less-than operator (-5 < -3)
* 305 – Correct evaluation of less-than-or-equal operator (6 <= 6 == true)
* 332 – Associativity of greater-than operator (3 > 7 >= -1)
* 112 – Correct evaluation of greater-than-or-equal operator (5 >= 5)
* 303 – Precedence of less-than operator (7 < 3 != true)
* 393 – Associativity of comparison/equals operator (6 == 3 != true)
* 1243 – Precedence among lower-level arithmetic and relational operators (2 + 7 == 3 \* 3)
* 1281 – Precedence among lower-level arithmetic and relational operators (3 - -5 != 3 + 5)
* 1132 – Precedence among lower-level arithmetic and high-level relational operators (5 – 9 <= 9 – 5)
* 1111 – Precedence among lower-level arithmetic and high-level relational operators (8 – 6 < 8 + 2)
* 1291 – Precedence among high-level arithmetic and lower-level relational operators (-3 \* -4 != 6 \* 2)
* 1272 – Precedence among high-level arithmetic and lower-level relational operators (7 % 3 == 7 / 3)
* 1225 – Precedence among high-level arithmetic and high-level relational operators (8 / 5.0 >= 8 / 5)
* 1211 – Precedence among high-level arithmetic and high-level relational operators (3 \* 5 > 5 \* 3)
* 351 – Associativity of less-than-or-equal operator (6 <= 6 < 2)
* 314 – Precedence of greater-than-or-equal operator (3.6 >= 6.3 == false)
* 317 – Precedence of greater-than operator (true != 3 > 6)
* 1413 – Precedence of parenthesis (3 < 6 != (6 > 3))
* 1467 – Associativity of parenthesis ((5 == 9) != (9 != 5))

**Logical Expressions**

* 253 – Precedence of and with respect to other logical operators (false || true && false && true)
* 135 – Correct evaluation of and operator whose operands are arithmetic expressions (3 - 6 && 6 – 3)
* 355 – Associativity of and operator (true && false && true)
* 205 – Short circuit evaluation of and operator whose operands are arithmetic expressions (3 \* 0 && 3 + 0)
* 555 – Precedence of or operator with respect to other logical operators (false || true && false || false)
* 430 – Correct evaluation of or operator whose operands are arithmetic expressions (3 \* 0 || 0 – 3)
* 486 – Short circuit evaluation of or operator (true || false && true)
* 506 – Short circuit evaluation of or operator whose operands are arithmetic expressions (6 + 3 || 6 – 3)
* 655 – Associativity of or operator (false || true || false)
* 783 – Precedence of not operator (true && ! false)
* 760 – Correct evaluation of not operator applied to non-relational expression (! 3 – 3)
* 792 – Associativity of not operator (! ( 12 > 6 ) || ! ( 12 < 6 ))
* 913 – Precedence of Parenthesis (false || (false || true) && false)
* 959 – Associativity of Parenthesis (true && (true && (true || false)))

**Assignment Expressions**

* 130 – Simple self-referential assignment (var1 = var1 + 5)
* 502 – Correct evaluation of prefix increment operator (var2 = ++ var1)
* 702 – Correct evaluation of postfix increment operator (var2 = var1 ++)
* 255 – Correct evaluation of compound arithmetic operator (var1 += var2)
* 802 – Correct evaluation of postfix decrement operator (var2 = var1--)
* 602 – Correct evaluation of prefix decrement operator (var2 = -- var1)
* 210 – Widening coercion in simple assignment (double var1 = 3 + 5)
* 528 – Precedence of prefix increment operator (5 + ++ var1)
* 300 – Precedence of compound assignment (var1 += 3 + 5)
* 732 – Precedence of postfix increment operator (5 <= var1 ++)
* 237 – Narrowing coercion in simple assignment (int var1 = 3 \* 2.0)
* 630 – Precedence of prefix decrement operator (5 < -- var1)
* 455 – Narrowing coercion in compound assignment (int var1 += 5.0)
* 836 – Precedence of postfix decrement operator (var2 += var1 --)
* 182 – Associativity of simple assignment (var1 = var2 = 5)
* 366 – Associativity of compound assignment (var1 += var2 += 5)
* 550 – Associativity of prefix operator (++ ++ var1)

**Selection statements**

* 107 – Condition of if-else is a relational expression, evaluates to true
* 134 – Condition of if-else is a relational expression, evaluates to false
* 404 – Cascading nested if-else statements (nesting in only one clause)
* 429 – Classification nested if-else (nesting in both clauses)
* 502 – Multiple if-else statements
* 612 – Condition of if statement is a relational expression. Evaluates to true
* 634 – Condition of if statement is a relational expression. Evaluates to false
* 653 – Condition of if statement is a declaration, evaluates to true
* 702 – Condition of if statement is an arithmetic expression, evaluates to true
* 727 – Condition of if statement is an arithmetic expression, evaluates to false
* 826 – Nested if statements
* 853 – Multiple if statements

**Switch statements**

* 106 – Condition matches one of the cases
* 136 – Condition matches default case
* 151 – Condition matches no case
* 185 – Condition is an expression
* 202 – Condition is not an integer expression
* 303 – No statement for a case
* 227 – Case value is not an integer expression
* 329 – The body has no break, but has fall-through statements
* 257 – Case value is not a constant expression
* 354 – The body has no break and no statements
* 402 – Nested switch statements
* 502 – Multiple switch statements

**While loops**

* 331 – Condition is a relational expression
* 377 – Condition where sentinel value also changes on each iteration
* 555 – Variable is updated before being printed in the body of the loop
* 658 – Code executed after the loop exits
* 108 – Loop is never iterated
* 128 – Loop iterates only once
* 752 – Nested independent loops
* 726 – Nested dependent loops
* 688 – Multiple dependent loops

**For loops**

* 331 – Condition is a relational expression
* 376 – Condition where sentinel value also changes on each iteration
* 430 – Body of the loop is a simple statement
* 478 – Variable is updated before being printed in the body of the loop
* 654 – Code is executed after the loop exits
* 103 – Loop is never iterated
* 133 – Loop is iterated only once
* 752 – Nested independent loops
* 727 – Nested dependent loops
* 690 – Multiple dependent loops

**Do-while loops**

* 204 – Condition is a relational expression
* 305 – Condition involves and operator
* 527 – Multiple independent loops
* 227 – Condition is an arithmetic expression
* 328 – Condition involves or operator
* 501 – Multiple dependent loops
* 460 – Condition where sentinel value also changes on each iteration
* 353 – Condition compares against multiple discrete sentinel values
* 578 – Nested independent loops
* 252 – Condition is an assignment expression
* 378 – Condition excludes multiple discrete sentinel values
* 551 – Nested dependent loops
* 104 – Loop iterates only once
* 403 – Condition is negated
* 477 – Condition is empty

**Advanced loop concepts**

* 111 – A break statement inside a loop
* 303 – Infinite loop - Condition variable is not updated in the loop
* 506 – Multiple dissimilar loops, each dependent on the previous loop
* 329 – Infinite loop - Condition variable is incorrectly updated in the loop
* 211 – Correct use of continue statement
* 158 – break statement not inside a loop or switch statement
* 485 – Condition is a constant - either always true or always false
* 556 – Multiple dissimilar nested loops, inside loop dependent on outside loop
* 357 – Infinite loop - Value against which condition variable is compared is also modified
* 131 – Break inside a nested loop
* 234 – continue statement not inside a loop
* 608 – Accumulator variable used in a loop
* 385 – Infinite loop - Assignment used in condition

**Debugging functions (Function/Debug)**

* 630 – Number of actual parameters in the call not equal to the number of formal parameters in the definition
* 436 – Value returned by the function is ignored by the caller
* 211 – return statement missing in the definition of a function with non-void return type
* 654 – Data type of the actual parameter in the call is incompatible with the data type of the formal parameter in the definition
* 460 – Caller tries to use the value returned by a function with void return type
* 236 – Data type of the value in the return statement is incompatible with the return type of the function
* 279 – Formal parameter is re-declared as a local variable in the function
* 261 – return statement is not the last statement in the function
* 410 – Function is called before it is defined/prototyped

**Tracing functions (Function/Output)**

* 505 – One parameter passed by value, actual parameter is a variable
* 585 – One parameter passed by value, actual parameter is a literal constant
* 311 – Function call is part of an expression
* 535 – Multiple parameters passed by value, actual parameters are variables
* 605 – One parameter passed by value, actual parameter is an expression
* 129 – Multiple return statements in a function
* 335 – Calls to multiple functions
* 106 – Same name used for variables in two different functions
* 354 – Multiple calls to the same function
* 385 – Chained function calls

**Array**

* 205 – Initialization of all the elements of an array
* 311 – Assignment to random elements of an array
* 429 – Referencing elements of an array using a for loop
* 331 – Assignment to elements of an array using a for loop
* 403 – Referencing random elements of an array
* 380 – Self-referential assignment to elements of an array (a[i] = a[i + 1])
* 456 – Referencing an element of an array before initializing its value
* 502 – Array subscript is an expression
* 527 – Accessing an element outside the bounds of the array
* 605 – Entire array is passed as parameter to a function, where it is referenced
* 628 – Entire array is passed as parameter to a function, where it is assigned
* 654 – An element of an array is passed as parameter to a function
* 681 – Data type of the actual parameter in the call is incompatible with the data type of the formal parameter in the definition
* 255 – Incomplete initialization of an array

**OOP**

* 103 – Client accessing public variable of an object
* 227 – Member function accessing protected variable of an object
* 328 – Client cannot call private function of a class
* 402 – Constructor is called when an object is created (no parameters passed)
* 128 – Client cannot access protected variable of a class
* 253 – Member function accessing private variable of an object
* 354 – Non-static function of a class cannot be called with respect to the class
* 428 – Constructor is called when an object is created (parameters passed)
* 153 – Client cannot access private variable of a class
* 203 – Member function accessing public variable of an object
* 303 – Client cannot call protected function of a class
* 454 – Number of actual parameters in the call not equal to the number of formal parameters in the definition
* 505 – Client calls a selector with respect to an object
* 279 – The variable is not a static member of the class and cannot be accessed with respect to the class
* 603 – Client calls a mutator with respect to an object
* 476 – Data type of the actual parameter in the call is incompatible with the data type of the formal parameter in the definition
* 701 – Destructor is called when object is deallocated
* 553 – Selector references a class variable before initializing it