Fill in each of the following:

|  |  |  |
| --- | --- | --- |
|  | **Expression** | **Your Answer:** |
|  | 2 + 3 \* 2.0 | EXAMPLE:   * Annotated with types: 2[int] + 3[int] \* 2.0[double] * Multiplication goes first * Convert 3[int] to 3.0[double] * 2[int] + 3.0[double] \* 2.0[double] * Do the multiplication: 2[int] + 6.0[double] * Addition goes next * Convert 2[int] to 2.0[double] * 2.0[double] + 6.0[double] * Do the addition * 8.0 [double] is the final result |
|  | 2 \* 6 / 3 | * Annotated with types 2[int] \* 6[int] / 3[int] * Multiplication goes first * Do the multiplication 2[int] \* 6[int] * Get 12[int] * Division goes next * Do the division 12[int]/3[int] * Final result: 4[int] |
|  | 20 – 6 / 2 | * Anotated with types 20[int] - 6[int] / 2[int] * Division goes first * Do the division 6[int]/2[int] * Get 3[int] * Subtraction goes next * Do 20[int] - 3[int] * Final result: 17[int] |
|  | 43 > 20 && 20 > 10 | * Anotated with types 43[int] > 20[int] &&[AND-gate] 20[int] > 10[int] * Do the most left operator first 43[int]>20[int] * Get: 'TRUE' * Do the second left most operator 20[int] > 10[int] * Get: 'TRUE' * Check 'Logic Gate': && * Criterion suffices * Final result: 'TRUE' |
|  | int x = 43;  int y = 20;  bool z = (x >= y && y < x); | * Anotate and store 43[int] in variable x[int] * Anotate and store 20[int] in variable y[int] * Anotate and store (x[int] >= y[int] &&[AND-gate] y[int] < x[int]) in variable z[bool] * Check criterion inside z[bool] * Anotate with types x[int] >= y[int] &&[AND-gate] y[int] < x[int] * Fill in missing variables: 43[int] >= 20[int] &&[AND-gate] 20[int]< 43[int] * Do the most left operator first 43[int] >= 20[int] * Get: 'TRUE' * Do the second most operator 20[int] < 43[int] * Get: 'TRUE' * Check 'Logic Gate': && * Criterion suffices * Get: 'TRUE' * Final result: int x = 43[int], int y = 20[int], z = 'True' |
|  | int x = 2;  int y = 2 \*x + 1; | * Anotate and store 2[int] in variable x[int] * Anotate and store (2[int] \* x[int] + 1[int]) in y[int] * Check criterion inside of y[int] * Anotate with types 2[int] \* x[int] + 1[int] * Fill in missing variables: 2[int] \* 2[int] + 1[int] * Do multiplication first 2[int] \* 2[int] * Get 4[int] * Addition goes next * 4[int] + 1[int] * Get 5[int] * Final result: int x = 2[int], int y = 5[int] |
|  | int x = 2;  int y = 2 \* (x + 1); | * Anotate and store 2[int] in variable x[int] * Anotate and store (2[int] \* (x[int] + 1[int]) in y[int] * Check criterion inside of y[int] * Anotate with types 2[int] \* (x[int] + 1[int]) * Fill in missing variables: 2[int] \* (2[int] + 1[int]) * Do parenthesis first 2[int] + 1[int] * Do addition next 2[int] + 1[int] * Get 3[int] * Do multiplication next 2[int] \* 3[int] * Get 6[int] * Final result: int x = 2[int], int y = 6[int] |
|  | int x = 2;  bool z = 2 \* x >= 10; | * Anotate and store 2[int] in variable x[int] * Anotate and store (2[int] \* x[int] >= 10[int]) in variable z[bool] * Check criterion inside of z[bool] * Anotate with types 2[int] \* x[int] >= 10[int] * Fill in missing variables: 2[int] \* 2[int] >= 10[int] * Do most left operator first * Do multiplication next 2[int] \* 2[int] * Get 4[int] * Check operator 4[int] >= 10[int] * Get 'FALSE' * Final result: int x = 2[int], bool z = 'FALSE' |