1. The programmer was having a bad day writing this program. Correct the errors so that there are no compile time errors. Make whatever assumptions you can about data types that need fixing up.

```cpp
#include <iostream>
const int TOTAL = 500;
int main () {
    using namespace std;
    double item’sCost;
    quantityOnHand = 42;
    double total_cost;
    cupon_discount int;
    const double AmountPaid;
    item’sCost = 14.99
    AmountPaid = 19.99;
    ...
```

2. Circle the variable names that are invalid C++ variable names. Do not circle ones that are legal, even though they might not represent the best naming convention.

   CostOfGoodsSold
   total Ticket Price
   R2D2
   3D-Movie
   distance Traveled
   cos
   fAbs
   Log
   qty_sold
   qty sold
   qtySold

3. Convert each of these formulas into a proper C++ statement.

   a. Force = mass times acceleration

   b. A = PI R^2 (area of a circle)
c. \[ total = \frac{n(n + 1)}{2} \]

d. \[ x = \sin(2\ \pi\ \text{theta}); \]

e. \[ z = \sqrt{x^2 + y^2} \]

f. \[ x = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \]

g. \[ x = \sqrt[4]{z} \]

4. Correct the errors in these C++ calculations.

a. \[ \text{cost} = \text{qty} \times \text{unitCost}; \quad // \text{cost is qty times unitCost} \]

b. \[ \text{total} = \text{total} + + \text{tally}; \quad // \text{add tally to total} \]

c. \[ \text{sum} + 1 = \text{sum} \quad // \text{add one to sum} \]

d. \[ \text{root} = \sqrt{\text{x} \times \text{x} + \text{y} \times \text{y}}; \]
    \[ // \text{root is the square root of x squared and y squared} \]

e. \[ \text{xy} = \text{Pow}(\text{x}, \text{y}); \quad // \text{calculate x raised to the y\text{th} power} \]

f. \[ \text{count} + 1; \quad // \text{increment count} \]
5. The equation to be solved is shown below.

\[
\text{percent1} = \frac{\text{salesTotal1}}{\text{salesTotal1} + \text{salesTotal2}} \times 100
\]

Assuming all variables are **doubles**, which of the following correctly calculates the percentage? Next, assuming all variables are **integers**, which of the following correctly calculates the percentage? Indicate which work for doubles and which work for integers by placing an I or a D before each letter.

a. \( \text{percent1} = \frac{\text{salesTotal1}}{\text{salesTotal1} + \text{salesTotal2} \times 100}; \)

b. \( \text{percent1} = \frac{\text{salesTotal1}}{\text{salesTotal1} + \text{salesTotal2} \times 100}; \)

c. \( \text{percent1} = \frac{\text{salesTotal1}}{\text{salesTotal1} + \text{salesTotal2}} \times 100; \)

d. \( \text{percent1} = \frac{((\text{salesTotal1})}{\text{salesTotal1} + \text{salesTotal2}) \times 100; \)

e. \( \text{percent1} = \frac{\text{salesTotal1} \times 100}{\text{salesTotal1} + \text{salesTotal2}; \)

f. \( \text{percent1} = \frac{\text{salesTotal1} \times 100}{\text{salesTotal1} + \text{salesTotal2}); \)

6. Show the precise output from the following series of **cout** instructions. Assume these are the initial values of the variables. Assume the **ios::fixed** has been set along with **ios::showpoint**. Assume that the precision has been not yet been set to any value.

```cpp
int x = 123;
double z = 42.35353;

a. cout << setw (6) << x << x;

b. cout << x << setw (6) << x;
```
7. For each of these short calculations, show the result that is displayed. Assume that `ios::fixed` and `ios::showpoint` have been set on `cout` and that the precision is set to two decimal digits unless overridden.

a. 
```cpp
int x = 10;
int y = 4;
cout << x / y;
```

b. 
```cpp
int pennies = 123;
const int DIMES = 10;
int dimes;
dimes = pennies / DIMES;
pennies = pennies % DIMES;
cout << dimes << " " << pennies;
```

c. 
```cpp
double number = 25;
cout << sqrt (number);
```

d. 
```cpp
double num = 5;
```
double bignum;
bignum = pow (num, 2);
cout << setprecision (0) << bignum;