#Description: rounding issues notes

Root Cause

In solidity there are no floating point numbers (no decimals)

9 / 10 = 0.9

Solidity: 9 / 10 = 9

ERC20 Decimals

WETH 18 decimals

1×10^{18} = 1 WETH

USDC 6 decimals

1×10^{6} = 1 USDC = 1_000_000

$\frac{1 \times 10^{6}}{2} = 5 \times 10^{5} = 0.5 \times 10^{6}$

If an erc20 token has zero decimals

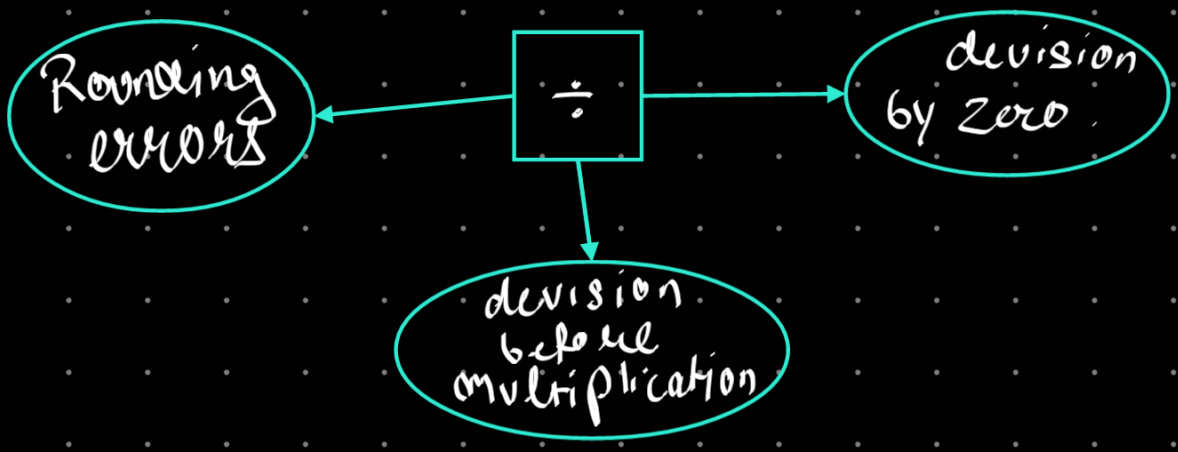
USDgg 0 decimals

1×10^{0} = 1 usdgg

1 usdgg / 2 = 0

Common Issues

Common Issues



Finding 1

Long Position PnL

PnL = (Position size in tokens × current market price − position size in tokens × avg Entry Price)

Short Position PnL

PnL = (Position size in tokens × avg Entry Price − Position size in tokens × current market price)

```
// Calculate average price for position
uint256 updatedAvgPrice = (
    userPosition.positionSize * userPosition.avgPrice + userOrder.orderSize * _executionPrice
) / (userPosition.positionSize + userOrder.orderSize);
```

If this numerator is not completely divisible by denominator then remainder will be scrapped. which means avgPrice will stay the same even if marketprice has increased. Some attacker could leverage this for profit.

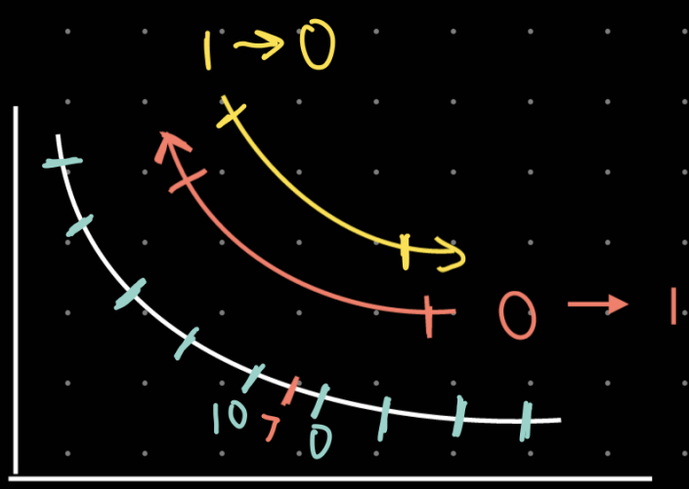
Mitigation

round up division for longs

Use rounding up
round down for shorts

Finding 2

Ticks



Round to 10 if 0 → 1

Round to 0 if 1 → 0

```
function roundAhead(
    int24 tick,
    int24 tickSpacing,
    bool zeroForOne
) internal pure returns (
    int24 roundedTick
) {
    roundedTick = tick / tickSpacing * tickSpacing;
    if (roundedTick == tick) return tick;
    /// @dev - rounding down only needed if negative
    if (zeroForOne && roundedTick < 0)
        roundedTick += tickSpacing;
    /// @dev - rounding up only needed if positive
    else if (!zeroForOne && roundedTick > 0)
        roundedTick -= tickSpacing;
}
```

roundedTick = 17

17 / 10 = 1 × 10 = 10

but if rounding tick < tickspacing

7 / 10 = 0 × 10 = 0

rounded tick = 0 if tickspacing is greater

So it will round down even with 0 → 1.

Mitigation

```
function roundAhead(
    int24 tick,
    int24 tickSpacing,
    bool zeroForOne
) internal pure returns (
    int24 roundedTick
) {
    roundedTick = tick / tickSpacing * tickSpacing; // -17 / 10 = -1 * 10 = -10
    if (roundedTick == tick) return tick; // 10 / 10 * 10 = 10
    if (zeroForOne && (roundedTick > 0 || (roundedTick == 0 && tick > 0))) roundedTick += tickSpacing;
    else if (!zeroForOne && (roundedTick < 0 || (roundedTick == 0 && tick < 0))) roundedTick -= tickSpacing;
}
```

account for when roundedTick < and == 0.

if ÷ think about truncation

- is truncation handeled correctly
- is it expected
- what will happen if i provide small values
- edgecases with negative numbers