pragma solidity ^ 0.5.1;

/// @title Strategy contract

/// @author Andrea Bruno 585457

/// @notice This abstract contract describes what a strategy should implement.

/// @dev The following comments are written using the Solidity NatSpec Format.

contract Strategy {

/// @notice This function compute the price.

/// @param \_actualPrice is the current price of the item

/// @param \_deltaBlocks the blocks elapsed.

/// @return The updated current price.

function getPrice(uint \_actualPrice, uint \_deltaBlocks) public view returns(uint);

}

/// @title NormalStrategy contract

/// @notice This contract implement a linear decrease function.

/// @dev This contract extends `Strategy`

contract NormalStrategy is Strategy {

function getPrice(uint \_actualPrice, uint \_deltaBlocks) public view returns(uint) {

uint tmp = \_actualPrice - \_deltaBlocks;

//in case of underflow

if (tmp > \_actualPrice) return 0;

else return tmp;

}

}

/// @title FastStrategy contract

/// @notice This contract implement a linear decrease function, twice faster than `NormalStrategy`

/// @dev This contract extends `Strategy`

contract FastStrategy is Strategy {

function getPrice(uint \_actualPrice, uint \_deltaBlocks) public view returns(uint) {

uint tmp = \_actualPrice - (2 \* \_deltaBlocks);

//in case of underflow

if (tmp > \_actualPrice) return 0;

else return tmp;

}

}

/// @title SlowStrategy contract

/// @notice This contract implement a linear decrease function, twice slower than `NormalStrategy`

/// @dev This contract extends `Strategy`

contract SlowStrategy is Strategy {

function getPrice(uint \_actualPrice, uint \_deltaBlocks) public view returns(uint) {

uint tmp = \_actualPrice - (\_deltaBlocks / 2);

//in case of underflow

if (tmp > \_actualPrice) return 0;

else return tmp;

}

}