Problem 1: Liquidations Problem

Part (a): Calculating the Health Factor

From given Information we can calculate these:

- Exchange rates:
  - 1 ETH = 1500 USDC
  - 1 ETH = 100 AXS → 1 AXS = 0.01 ETH

Since Bob holds 1000 USDC, we convert this to ETH:

1000 / 1500 = 0.6667 ETH

Applying the collateral factor of 0.9:

0.6667 × 0.9 = 0.6 ETH

- Debt in ETH: 0.2 ETH
- Debt in AXS (converted to ETH): 20 × 0.01 = 0.2 ETH
- Total Debt: 0.2 + 0.2 = 0.4 ETH

Health Factor = BorrowCapacity / value(TotalDebt)

= 0.6 / 0.4 = 1.5

Since 1.5 > 1, Bob's position is healthy and does not need liquidation.

Part (b): When the AXS Exchange Rate Changes

New AXS exchange rate: 1 AXS = 1/48 ETH ≈ 0.02083 ETH

New AXS debt in ETH: 20 × 0.02083 = 0.4167 ETH

New total debt: 0.2 + 0.4167 = 0.6167 ETH

Step 2: Compute New Health Factor

0.6 / 0.6167 ≈ 0.973

Since 0.973 < 1, Bob's collateral must be liquidated.

Part (c): How Much USDC Needs to Be Liquidated?

A liquidator is willing to clear Bob's ETH debt in exchange for USDC collateral at a rate of 1520 USDC/ETH.

To restore a healthy position, let x be the amount of ETH debt to be cleared.

New debt after liquidation: $0.6167 - x$

Solving for x:

$0.6167 – x <= 0.6$

$=> x >= 0.0167$ ETH

Choose $x = 0.0167$ ETH (the min value of x)

USDC to be liquidated = $0.0167 \times 1520 \approx 25.38$ USDC

Rounding up, Bob will lose at least 26 USDC from his collateral to restore a healthy position.

Problem 2: Slippage

We are tasked with showing that slippage (s) is always positive and approximates x/X, assuming X >> x. Below is the step-by-step derivation:

We are given the following formulas:

1. Constant Product Formula (Uniswap invariant):
 $x * y = k$

2. Amount of Y tokens Alice needs to send to buy x amount of X tokens:
 $y = Y * x / (X - x)$

3. Exchange rate Alice gets from Uniswap:
 $y / x = Y / (X - x)$

4. Open market exchange rate:
 $p = Y / X$

5. Slippage definition:
 $s = ((y / x) - p) / p$

$s = ((Y / (X - x)) - (Y / X)) / (Y / X)$

$s = (Y * (1 / (X - x) - 1 / X)) / (Y / X)$

$s = ((1 / (X - x) - 1 / X)) * X$

Using the identity $(1 / (X - x) - 1 / X) = x / (X * (X - x))$:

$s = X * (x / (X * (X - x))) = x / (X - x)$

If X >> x, then $X - x \approx X$, so we can approximate:

$s \approx x / X$

Since $x, X > 0$, it follows that $s > 0$, meaning the exchange rate on Uniswap is always worse than on the open market.

However, as the liquidity pool size X increases, s decreases, meaning larger liquidity pools reduce slippage.

Problem 3: Sandwich Attacks on Uniswap

Part (a): Effect of Sam's Tx1 on Alice's Transaction

Initially, the pool has reserves x tokens of type X and y tokens of type Y, with $k = xy$.
Alice's transaction Tx sends αx tokens of type X, increasing the X reserves to $x + \alpha x$. Uniswap maintains the constant product $xy = k$, so Alice receives:

$$y' = (xy) / (x + \alpha x)$$

Now, when Sam front-runs Alice with Tx1, he sends βx tokens of type X to the pool first. This increases the X reserves to $x + \beta x$ and reduces the Y reserves accordingly:

$$y1 = (xy) / (x + \beta x)$$

Then, Alice's Tx executes, sending αx more tokens of type X. The new X reserves become:

$$x + \beta x + \alpha x$$

and Uniswap ensures the constant product:

$$y' = (xy) / (x + \beta x + \alpha x)$$

The amount of Y Alice receives is now:

$$y' - y1 = (xy) / (x + \beta x + \alpha x) - (xy) / (x + \beta x)$$

Since $x + \beta x + \alpha x > x + \alpha x$, Alice now receives fewer Y tokens than she would have without Sam's front-running.

Part (b): How Does Sam's Profit Depend on β?

Sam's profit comes from the arbitrage between his front-run and back-run transactions.
In Tx1, Sam increases the X reserves by βx, which reduces the Y reserves and makes Y more expensive.
Alice then swaps her X, getting fewer Y tokens than before.
In Tx2, Sam swaps back the Y he bought earlier, benefiting from the temporary price impact

he created.
Increasing β makes Alice's swap worse (for her), which means Sam can extract more profit up to a point.
However, if β is too large, Sam's Tx1 creates excessive slippage, reducing his own profit.

Conclusion: Sam's profit initially increases with β but can decrease if β is too large due to excessive price impact.

Part (c): Can Victor Mount the Attack Without Sam?

Yes! Since Victor, the validator, controls the transaction ordering, they can copy Sam's strategy.
However, Victor does not have X tokens to initiate the front-run.
If Victor has access to flash loans, they can temporarily borrow X tokens to execute Tx1 and repay in Tx2, keeping all profit for themselves.

Conclusion: If flash loans are available, Victor can replace Sam and execute the sandwich attack solo.

Part (d): Can Alice Use MEV-Boost to Protect Herself?

MEV-Boost allows Alice to submit her transaction privately to a trusted block builder instead of the public mempool.
This prevents Sam (or any searcher) from seeing her transaction in advance, blocking sandwich attacks.
If Alice uses MEV-Boost, the only party that can still manipulate the order is the builder itself.
However, if Alice picks a trusted, honest builder, she can avoid the attack altogether.

Conclusion: MEV-Boost can help protect Alice, but it depends on the integrity of the builder she selects.