A13: DeFi Builder

Deploy Your Own DEX

Prof. Joerg Osterrieder

Spring 2026

Assignment Overview

Learning Objectives

- Deploy two ERC-20 tokens and a SimpleDEX on Remix
- Add liquidity and execute swaps
- Observe how $x \times y = k$ determines prices
- Measure price impact across different trade sizes

Assignment Details

- Time: 75 minutes
- Format: Groups of 2–3 students
- Difficulty: Hard
- Points: 60 (+10 bonus)

Grading Breakdown

- DEX deployed + working: 30 pts
- Worksheet (prices): 15 pts
- Presentation: 15 pts
- Bonus (slippage/escrow): +10 pts

A13:

digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

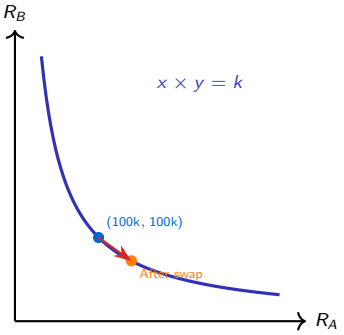
The Constant Product Formula

How AMMs Price Assets

- Pool holds reserves of two tokens: R_A and R_B
- Invariant: $R_A \times R_B = k$ (constant)
- To buy Token B, you add Token A
- Formula: $\text{out} = \frac{R_B \times \text{in}}{R_A + \text{in}}$

Key Insight

"No order book. No matching engine. Just a formula and two token reserves. The price emerges from the ratio of reserves."



A13:

digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

Step 1: Deploy Two ERC-20 Tokens

15 min — 8 pts

Steps

1. Open remix.ethereum.org
2. Create `MyToken.sol` and paste the ERC-20 code
3. Compile with Solidity 0.8.20+
4. Deploy **Token A**: custom name, symbol, supply = 1,000,000
5. Deploy **Token B**: different name/symbol, same supply
6. Copy both contract addresses

ERC-20 Functions Needed

- `balanceOf()` – check balance
- `approve()` – allow DEX to spend
- `transferFrom()` – used by DEX
- `transfer()` – direct sends

Common Mistake

"Don't forget to copy the token addresses after deploying! You need them for the DEX constructor."

A13:

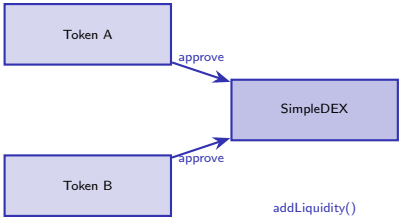
digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

Step 2: Deploy the SimpleDEX

10 min — 10 pts

Steps

1. Create `SimpleDEX.sol` in Remix
2. Deploy with constructor args: Token A address, Token B address
3. On Token A: `approve(DEX_addr, 500000)`
4. On Token B: `approve(DEX_addr, 500000)`
5. On DEX: `addLiquidity(100000, 100000)`
6. Verify: `reserveA` = 100000, `reserveB` = 100000



#1 Error: Forgot approve()

"Before addLiquidity or swap, you MUST call approve() on the input token to let the DEX spend your tokens."

A13:

digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

Step 3: Execute Swaps & Track Prices

25 min — 12 pts

Swap Sequence

1. **Swap 1 (Small):** 1,000 Token A → Token B
   - Expected out: ∼990 B
   - Price impact: ∼1%
2. **Swap 2 (Medium):** 10,000 Token A → Token B
   - Expected out: ∼8,919 B
   - Price impact: ∼11%
3. **Swap 3 (Large):** 50,000 Token A → Token B
   - Expected out: ∼27,978 B
   - Price impact: ∼44%
4. **Swap 4 (Reverse):** 10,000 Token B → Token A
   - Expected out: ∼22,326 A
   - Exploits reserve imbalance

For Each Swap, Record:

- Amount in / amount out
- Reserve A / Reserve B (after)
- $k = R_A \times R_B$
- Effective price

Price Impact

"Watch how the effective price degrades from 0.99 (small swap) to 0.56 (large swap). This is why liquidity depth matters!"

A13:

digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

Step 4: Complete the Worksheet

10 min — 15 pts

Worksheet Tasks

1. Fill in the price-tracking table (all 4 swaps)
2. Calculate effective price for each swap
3. Verify k stays approximately constant
4. Answer analysis questions:
   - Why does a larger swap give worse prices?
   - Does $x \times y = k$ hold exactly?
   - What happened with the reverse swap?

Expected Results Summary

Swap	Out	Price
1k A→B	990	0.99
10k A→B	8,919	0.89
50k A→B	27,978	0.56
10k B→A	22,326	2.23*

*A per B (reverse direction)

Why k Varies Slightly

Solidity uses integer division (rounds down). The "lost" fractions stay in the pool.

A13:

digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

Step 5: Presentation

15 min total (3 min/group)

Presentation Structure (3 min)

1. Token introduction (15 sec)
2. Price impact demo – show swap data (1 min)
3. Key observation about $x \times y = k$ (1 min)
4. Real-world connection to Uniswap (30 sec)
5. Q&A (15 sec)

Bonus Challenge (+10 pts)

- **Slippage Protection (+5):**
  - Add `_minOut` parameter to swap
  - `require(out >= _minOut)`
- **Simple Escrow (+5):**
  - Deploy L11 escrow contract
  - Demo deposit → confirm flow

Deliverables

- Working DEX on Remix
- Completed price worksheet
- 3-minute presentation

A13:

digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

Get Started

Assignment Page

digital-ai-finance.github.io/crypto-economics/assignments/A13_defi_builder/instructions.html

All Assignments

digital-ai-finance.github.io/crypto-economics/assignments/index.html

Open Remix and start deploying your tokens!

You have 75 minutes: 15 min tokens, 10 min DEX, 25 min swaps, 10 min worksheet, 15 min presentations.