שאלות:

1. בהינתן העברה:


```
{
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid":
"7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18",
      "vout": 0,
      "scriptSig" :
"3045022100884d142d86652a3f47ba4746ec719bbfbd040a570b1deccbb6498c75c4ae24cb02204
b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e3813[ALL]
0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fe423cc5412336376789d1
72787ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf",
      "sequence": 4294967295
    }
  ],
  "vout": [
      {
        "value": 0.01500000,
        "scriptPubKey": "OP_DUP OP_HASH160
 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG"
      },
      {
        "value": 0.08450000,
        "scriptPubKey": "OP_DUP OP_HASH160
 7f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG",
      }
    ]
  }
```


א. חשב את סכום העמלה.

ב. כמה satoshi הועברו בעסקה?

מספר העברה:

7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18

2.

א. כתוב סקריפט המחבר 2 מספרים. תאר בכל שלב במהלך הריצה את מצב המחסנית

ב. כתוב סקריפט המחזיר את תוצאת החישוב של $\frac{a^2}{b}$ כך שהסקריפט מתחיל כך:

$$[a\ b\ < fill\ in > \cdots]$$

יש לוודא ש$b \neq 0$ (במקרה שb הוא 0 נקבל שגיאה ועצירה ולא אינסוף).

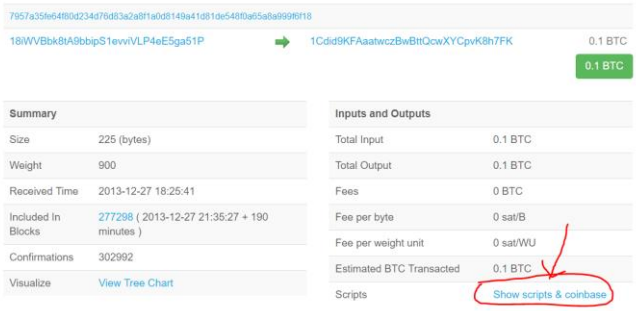
ג. כתוב סקריפט המחזיר את תוצאת החישוב של $\frac{a^2}{2} + \frac{c^3}{b}$ כך שהסקריפט מתחיל כך:

$$[a\ b\ c < fill\ in > \cdots]$$

יש לוודא ש$b \neq 0$ (במקרה שb הוא 0 נקבל שגיאה ועצירה ולא אינסוף).

3. התבוננו בהעברה כלשהי, כנסו לסקריפט והראו בסימולטור שהסקריפט אכן נותן תוצאה טובה (true). איזה סקריפטים צריך לצמד כדי שיעבדו?

https://bitcoin.stackexchange.com/questions/88340/bitcoin-script-execution-unsuccessful



.4

א. כתבו סקריפט שמבקש מספר חתימות.

ב. כאמור, בהעברה של חלק מסכום הביטקוין, שאר הUTXO צריך להיות מועבר חזרה אל כתובת בארנק – בארנקים מתקדמים ואפליקציות הארנק מבצע את ההעברה הזאת אוטומטית (שולח ל2 כתובות – לנמען ולמען).
אם כתבתי בעצמי את ההעברה ושכחתי להוסיף את כתובת הארנק ליתרה – מה יקרה לשאר הכסף?

.5 אליס מעביר לבוב 2 ביטקוין בהעברה תקנית וחותם עם המפתח הפרטי שלו. אם ינסה ג'ורג' לקחת את ההעברה הזו, לחתום אותה עם המפתח הפרטי שלו ולהעביר כסף הלאה – באיזה שלב הסקריפט יכשל? תאר את מצב המחסנית בכל שלב.

.6

א. מצא בקוד הפתוח של ביטקוין (interpreter) את פעולת אימות החתימה checksig.
הסבר איפה בדיוק הפעולה מתרחשת ואילו ארגומנטים מושווים.

ב. כל הבדיקות עבור העברות מסתכמות ב9 שגיאות. מנה אותן (היעזר בקוד המקור כדי לבדוק מהן 9 השגיאות האפשריות **(עבור העברות בלבד)**

תשובות:

.1 Fees = 0.0005. סכום מועבר ← 0.015. total input: 0.1
https://www.blockchain.com/btc/tx/7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18

.2

א. https://siminchen.github.io/bitcoinIDE/build/editor.html

ב. {a b op_dup op_verify op_swap op_dup op_mul op_swap op_div}

ג. {a b c op_dup op_mul op_swap op_dup op_verify op_div op_swap op_2div op_add}

.3 לא הצלחתי :
https://bitcoin.stackexchange.com/questions/88340/bitcoin-script-execution-unsuccessful

דוגמא לסקריפט שעובר:
304502203f004eeed0cef2715643e2f25a27a28f3c578e94c7f0f6a4df104e7d163f7f8f022100b8b248c1cfd8f77a0365107a9511d759b7544d979dd152a955c867afac0ef78601
044d05240cfbd8a2786eda9dadd520c1609b8593ff8641018d57703d02ba687cf2f187f0cee2221c3afb1b5ff7888caced2423916b61444666ca1216f26181398c OP_DUP OP_HASH160 2E67490797078511CFDE499434ED6A564AD41EF0 OP_EQUALVERIFY OP_CHECKSIG

.4

א. <a signature> <a pubkey> <b signature> <b pubkey> OP_DUP OP_HASH160 <a public key hash> OP_EQUAL OP_CHECKSIG OP_DUP OP_HASH160 <b public key hash> OP_EQUAL OP_CHECKSIG

ב. כל שאר הכסף יועבר בתור עמלה כי כך בדיוק עובדת עמלה, כסף שמוכנס בinput פחות ה amount בoutput.

5. https://www.youtube.com/watch?v=ir4dDCJhdB4 (43:30 דקה)

6.

א.

```
case OP_CHECKSIG:
case OP_CHECKSIGVERIFY:
{
    // (sig pubkey -- bool)
    if (stack.size() < 2)
        return set_error(serror, SCRIPT_ERR_INVALID_STACK_OPERATION);

    valtype& vchSig    = stacktop(-2);
    valtype& vchPubKey = stacktop(-1);

    // Subset of script starting at the most recent codeseparator
    CScript scriptCode(pbegincodehash, pend);

    // Drop the signature, since there's no way for a signature to sign itself
    scriptCode.FindAndDelete(CScript(vchSig));

    if (!CheckSignatureEncoding(vchSig, flags, serror) || !CheckPubKeyEncoding(vchPubKey, flags, serror)) {
        //serror is set
        return false;
    }
    bool fSuccess = checker.CheckSig(vchSig, vchPubKey, scriptCode);

    popstack(stack);
    popstack(stack);
    stack.push_back(fSuccess ? vchTrue : vchFalse);
    if (opcode == OP_CHECKSIGVERIFY)
    {
        if (fSuccess)
            popstack(stack);
        else
            return set_error(serror, SCRIPT_ERR_CHECKSIGVERIFY);
    }
}
break;
```

```
template <class T>
bool GenericTransactionSignatureChecker<T>::CheckSig(const std::vector<unsigned char>& vchSigIn, const std::vector<unsigned char>& vchPubKe
{
    CPubKey pubkey(vchPubKey);
    if (!pubkey.IsValid())
        return false;

    // Hash type is one byte tacked on to the end of the signature
    std::vector<unsigned char> vchSig(vchSigIn);
    if (vchSig.empty())
        return false;
    int nHashType = vchSig.back();
    vchSig.pop_back();

    uint256 sighash = SignatureHash(scriptCode, *txTo, nIn, nHashType, amount, sigversion, this->txdata);

    if (!VerifySignature(vchSig, pubkey, sighash))
        return false;

    return true;
}
```

המפתח הציבורי והחתימה מושווים (עם גיבוב החתימה).

ב. זהו הסיווג בקוד עבור שגיאות של העברות:

```
inline bool IsTransactionReason(ValidationInvalidReason r)
{
    return r == ValidationInvalidReason::NONE ||
           r == ValidationInvalidReason::CONSENSUS ||
           r == ValidationInvalidReason::RECENT_CONSENSUS_CHANGE ||
           r == ValidationInvalidReason::TX_NOT_STANDARD ||
           r == ValidationInvalidReason::TX_PREMATURE_SPEND ||
           r == ValidationInvalidReason::TX_MISSING_INPUTS ||
           r == ValidationInvalidReason::TX_WITNESS_MUTATED ||
           r == ValidationInvalidReason::TX_CONFLICT ||
           r == ValidationInvalidReason::TX_MEMPOOL_POLICY;
}
```

קיים הסבר באינומרציה עבור כל אחת מהשגיאות:

```
enum class ValidationInvalidReason {
    // txn and blocks:
    NONE,                    //!< not actually invalid
    CONSENSUS,               //!< invalid by consensus rules (excluding any below reasons)
    /**
     * Invalid by a change to consensus rules more recent than SegWit.
     * Currently unused as there are no such consensus rule changes, and any download
     * sources realistically need to support SegWit in order to provide useful data,
     * so differentiating between always-invalid and invalid-by-pre-SegWit-soft-fork
     * is uninteresting.
     */
    RECENT_CONSENSUS_CHANGE,
    // Only blocks (or headers):
    CACHED_INVALID,          //!< this object was cached as being invalid, but we don't know why
    BLOCK_INVALID_HEADER,    //!< invalid proof of work or time too old
    BLOCK_MUTATED,           //!< the block's data didn't match the data committed to by the PoW
    BLOCK_MISSING_PREV,      //!< We don't have the previous block the checked one is built on
    BLOCK_INVALID_PREV,      //!< A block this one builds on is invalid
    BLOCK_TIME_FUTURE,          //!< block timestamp was > 2 hours in the future (or our clock is bad)
    BLOCK_CHECKPOINT,        //!< the block failed to meet one of our checkpoints
    // Only loose txn:
    TX_NOT_STANDARD,          //!< didn't meet our local policy rules
    TX_MISSING_INPUTS,        //!< a transaction was missing some of its inputs
    TX_PREMATURE_SPEND,       //!< transaction spends a coinbase too early, or violates locktime/sequence locks
    /**
     * Transaction might be missing a witness, have a witness prior to SegWit
     * activation, or witness may have been malleated (which includes
     * non-standard witnesses).
     */
    TX_WITNESS_MUTATED,
    /**
     * Tx already in mempool or conflicts with a tx in the chain
     * (if it conflicts with another tx in mempool, we use MEMPOOL_POLICY as it failed to reach the RBF threshold)
     * TODO: Currently this is only used if the transaction already exists in the mempool or on chain,
     * TODO: ATMP's fMissingInputs and a valid CValidationState being used to indicate missing inputs
     */
    TX_CONFLICT,
    TX_MEMPOOL_POLICY,        //!< violated mempool's fee/size/descendant/RBF/etc limits
};
```