מטלה - ביטקוין

יש לענות על שאלה אחת לבחירתכם.

שאלה 1: ארנקים ועסקאות - תרגיל מעשי

[ע"פ יואב הניג]

א. פתחו שני ארנקי testnet עפ"י ההוראות בסרטון: https://www.youtube.com/watch?v=LfNE29AZ9I0

ב. מיצאו "ברז" (faucet) עבור רשת הניסוי (testnet), והשיגו משם ביטקוין.

ג. העבירו קצת כסף מארנק 1 לארנק 2.

ד. מיצאו את העיסקה שלכם בשרשרת הבלוקים של רשת הניסוי, למשל כאן:

`https://live.blockcypher.com/btc-testnet`

ה. כמה קלטים (inputs) יש בעיסקה שלכם? כמה פלטים (outputs)? מדוע?

ו. צרפו לפתרון השאלה, צילום מסך המראה את העיסקה שלכם בארנק ובשרשרת הבלוקים.

- **פתרון**: יצחק ונחום.

שאלה 2: קביעת רמת הקושי

לצורך השאלה, הניחו שקיימת המחלקה הבאה, המייצגת בלוק בשרשרת-הבלוקים:

```
struct Block {
    //... more methods
    long timestamp();    // time of block creation; seconds since 1/1/2008
    double difficulty();   // level of difficulty in block creation time.
    Block* previous();  // the preceding block in the chain.
    //... more methods
};
```

רמת הקושי של כריית בלוק צריכה להתעדכן בערך פעם בשבועיים. כיתבו פונקציה שאפשר להריץ פעם בשבועיים על-מנת להעריך את רמת הקושי הדרושה על-מנת שבלוק ייווצר בממוצע כל 10 דקות.

הפונקציה מקבלת קישור לבלוק האחרון בשרשרת, ומחשבת את רמת הקושי הדרושה בהתאם לקצב היצירה של הבלוקים מהשבועיים האחרונים. היעזרו בנוסחה הבאה, המחשבת את קצב היצירה הצפוי לפי רמת הקושי:

https://en.bitcoin.it/wiki/Difficulty#How_soon_might_I_expect_to_generate_a_block.3F

כותרת הפונקציה שאתם צריכים לממש:

```
double newDifficulty(Block* lastBlock);
```

- **פתרון**:

פתרון: הזמן הדרוש ליצירת בלוק (T) שווה לרמת הקושי (D) כפול קבוע כלשהו (C):

$$T = D*C$$

במשך שבועיים צריכים להיווצר כ-2016 בלוקים (14 ימים כפול 144 בלוקים ליום). לכן, דרך פשוטה לחישוב רמת הקושי היא:

- א. לחשב כמה זמן לקח ליצור את 2016 הבלוקים האחרונים ($T0$).
- ב. לקרוא מהבלוק את רמת הקושי האחרונה ($D0$).
- ג. בהתאם לנוסחה למעלה, לחשב את הקבוע $C = T0/D0$
- ד. הזמן הרצוי ליצירת 2016 בלוקים הוא שבועיים: $T1=14*86400$
- ה. מכאן, רמת הקושי החדשה היא:

$$D1 = T1/C = (T1/T0)*D0$$

עכשיו נשאר רק לכתוב את הפונקציה:

```
double newDifficulty(Block* lastBlock) {
```

```
    // a. calculate how much it took to create last 2016 blocks:
    long now = lastBlock->timestamp();
    for (int i=0, Block* b=lastBlock; i<2016; ++i)
         b = b->previous();
    long then = b->timestamp();
    long T0 = now-then;   // in seconds

    // b. read the most recent difficulty:
    long D0 = lastBlock->difficulty();

    // c. calculate the required time:
    long T1 = 14*86400;    // num of seconds in two weeks
    return (T1/T0)*D0;
}
```

שאלה 3: בדיקת תקינות עסקאות בשרשרת בלוקים

המטבע SimpleCoin הוא מטבע חדש (דמיוני), הדומה לביטקוין אבל הרבה יותר פשוט:

- בכל בלוק בשרשרת יש רק עיסקה אחת;
- הסכום של כל עיסקה הוא מטבע אחד בדיוק;
- לכל עיסקה יש רק נ.מ.ע.ן אחד (לא מפצלים מטבע לכמה נמענים שונים).

ישנם שני סוגי בלוקים: בלוק שבו נוצר מטבע חדש (למשל כשכר לכו.ר.ה), ובלוק שבו מועבר מטבע שנוצר קודם. כל בלוק בשרשרת מיוצג ע"י המחלקה הבאה:

```
class Block {
      Block previous;
   // קישור לבלוק הקודם בשרשרת. אם הבלוק הנוכחי הוא הראשון //
 // בשרשרת (נוצר הכי מוקדם), אז השדה מכיל בלוק ריק - null. //
      Block input;
   // הבלוק הכולל את הקלט לעיסקה זו - העיסקה שבה נוצר //
   // או נמסר המטבע שמשלמים בעיסקה זו. אם אין קלט כי המטבע //
   // נוצר בעיסקה הנוכחית, אז שדה זה מכיל בלוק ריק - null //
      Key receiver;  // המפתח הציבורי שאליו מועבר המטבע בעיסקה זו.
};
```

עיסקה חדשה מיוצגת ע"י המחלקה הבאה:

```
class Transaction {
      Block input;  // ראו הסבר למעלה
      Key receiver; // ראו הסבר למעלה
      Signature signature;   // חתימה דיגיטלית של שולח העיסקה
```

```
};
```

חתימה דיגיטלית מיוצגת ע"י המחלקה הבאה:

```
class Signature {
      bool is_valid(Key signer); // מקבלת כקלט את המפתח הציבורי של החותם
                                 // מחזירה "אמת" אם"ם החתימה תקינה.
};
```

בנוסף, נתון המשתנה הגלובאלי `latest` המייצג את הבלוק **האחרון** בשרשרת (זה שנוצר הכי מאוחר).

א. כיתבו פונקציה הבודקת האם עיסקה נתונה היא חוקית או לא:

```
void check(Transaction tx);
```

ניתן להניח שזו עיסקה של העברת מטבע ולא של יצירת מטבע חדש - כלומר `tx.input!=null`.

- **פתרון:**

```
void check(Transaction tx) {
     // שלב א: נבדוק אם הקלט של העיסקה הנתונה כבר נוצל בעיסקה קודמת. //
      for (Block b=latest;  b!=null;  b=b.previous) {
           if (b.input == tx.input)
               throw DoubleSpendException;
      }
      // שלב ב: נבדוק מי הבעלים של המטבע המועבר בעיסקה הנתונה, ונוודא שהוא אכן חתם על העיסקה. //
      Key coin_owner =  tx.input.receiver;
      if (!tx.signature.is_valid(coin_owner))
            throw IllegalSignatureException;
      System.out.println("The transaction is valid");
}
```

ב. ציירו שרשרת בת 4 בלוקים, והדגימו עליה את פעולת הפונקציה מהסעיף הקודם. תנו דוגמאות עם תוצאות שונות.

- **פתרון**: נניח שהשרשרת נראית כך *(מהמאוחר אל המוקדם)*:

```
•     latest == block4;
•     block4 == {
•          previous: block3,
•          input: block3,
•          receiver: 0xDDDDDDDD
•     }
•     block3 == {
•          previous: block2,
•          input: block1,
•          receiver: 0xCCCCCCCC
•     }
•     block2 == {
•          previous: block1,
•          input: null,
```

```
        receiver: 0xBBBBBBBB
    }
    block1 == {
        previous: null,
        input: null,
        receiver: 0xAAAAAAAA
    }
```

- **דוגמה *(1)***. מריצים את *check* על העיסקה הבאה:

```
    tx == {
        input: block1,
        receiver: 0xEEEEEEEE,
        signature: [חתימה חוקית של CCCCCCCC]
    }
```

- כשהבלוק *b* בלולאה מגיע ל-*block3*, תיזרק חריגה *DoubleSpendException*, כי ה-*input* של *block3* שווה ל-*input* של *tx* *(הקלט כבר נוצל)*.
- **דוגמה *(2)***. מריצים את *check* על העיסקה הבאה:

```
    tx == {
        input: block2,
        receiver: 0xEEEEEEEE,
        signature: [חתימה חוקית של CCCCCCCC]
    }
```

- הלולאה תסתיים בלי חריגה כי אין אף בלוק בשרשרת שה-*input* שלו שווה ל-*block2* *(הקלט לא נוצל)*.
- *אבל, החתימה היא של CCCCCCCC, בעוד שהבעלים הנוכחי של המטבע הוא BBBBBBBB (המקבל ב-block2). לכן תיזרק חריגה* IllegalSignatureException.

שאלה 4: חישוב יתרה

כיתבו אלגוריתם המקבל כקלט מפתח ציבורי כלשהו, ומחשב את ה"יתרה" של בעל המפתח בביטקוין.

- **פתרון**:

שאלה 5: מתקפת פיני (על-שם Hal Finney שחשב עליה ראשון)

נניח שמישהו, נקרא לו "התוקף", רוצה להשתמש במטבע אחד פעמיים. הוא פועל לפי האלגוריתם הבא. הוא כורה בלוקים באופן רגיל; בכל בלוק שהוא כורה, כאחת מ-2000 העיסקאות בבלוק, הוא מכניס עיסקה אחת של מטבע א מכתובת ב לכתובת ג, כאשר שתי הכתובות שייכות לו. ברגע שהוא מצא בלוק, במקום לפרסם אותו מייד, הוא הולך לחנות, קונה חפץ ומשלם עליו במטבע א מכתובת ב לכתובת של המוכר. ברגע שהוא יוצא מהחנות, הוא שולח את הבלוק המאושר שמצא בצעד הראשון. הבלוק מאושר, העיסקה שעשה עם המוכר נדחית, המוכר מפסיד את הכסף, והתוקף קיבל חפץ בלי לשלם.

א. מה הסיכון שהתוקף לוקח? באיזה מקרה ההתקפה תיכשל, וכמה יפסיד התוקף במקרה זה?

ב. המוכר מחכה t דקות לפני מסירת החפץ. מה צריך להיות ערך החפץ כך שההתקפה תהיה כדאית?

ג. מה יכול המוכר לעשות כדי להגן על עצמו מהתקפה זו?

- **פתרון**: דניאל ואיתן