TESTS

constructor

1. *Require #1*

מערך wallets אינו ריק

2. *Require #2*

0<K<=N

3. *Function Test*

לבדוק שכל המשתנים מאותחלים בערכים נכונים

sendChallenge

1. *Require #1*

מי שקורא לפונקציה אינו נמצא בקבוצה - נזרקת שגיאה

2. *Require #2*

קוראים לפונקציה זו אחרי שכבר קראנו לה ועדיין לא ענו לאתגר (יש כבר challenge קיים במערכת) - נזרקת שגיאה

3. *Require #3*

קוראים לפונקציה זו עם סכום שהוא נמוך מהסכום הדרוש עבור הpenalty - נזרקת שגיאה

4. *Require #4*

שולחים אתגר למישהו שאינו נמצא בקבוצה - נזרקת שגיאה

5. *Require #5*

מי שקורא לפונקציה חסום משליחת challenge (מכיוון שהוא כבר שלח אחד וענו ל-challenge הזה לכן הוא צריך לחכות מס בלוקים מסויים שווה לBLOCKS_TO_BLOCK) - נזרקת שגיאה

6. *Normal behavior test*

הקורא לפונקציה נמצא בקבוצה, במערכת אין challenge פעיל כרגע, יש לו מספיק כסף לשלוח יחד עם הפונקציה עבור הpenalty, הוא לא חסום לשליחת challenge...

respondToChallenge

1. *Require #1*

בדיקה האם מי שקרא לפונקציה אינו שייך לקבוצה (**מלכתחילה**) או-
בדיקה האם מי שקרא לפונקציה **כבר לא נמצא** בקבוצה (כלומר הוצא ממנה כי לקח לו זמן להגיב)

2. *Require #2*

בדיקה האם מי שקרא לפונקציה נמצא בקבוצה אך אין challenge פעיל

3. *Require #3*

בדיקה האם יש challenge פעיל אבל מי שענה אינו מי שקיבל את ה-challenge

4. *Function Test*

שליחת challenge, בדיקת פרמטרים, ממתינים (BLOCKS_TO_RESPOND/2) בלוקים, עונים לchallenge ובדיקת פרמטרים.

tryToRemoveChallengedUser

1. *Require #1*

המשתמש לא שייך לקבוצה.

2. *Require #2*

קוראים לפונקציה עוד לפני שיש challenge ולכן אי אפשר להסיר אותו - זורק שגיאה

3. *Normal behavior test (with removal)*

התנהגות הצפויה של הפונקציה: קוראים לפונקציה sendChallenge על יוזר ספציפי ואז קוראים לפונקציה שבודקים על אותו יוזר **<u>אחרי שעבר</u>** מס׳ הבלוקים BLOCKS_TO_RESPOND

4. *Normal behavior test (without removal)*

אותה בדיקה כמו בסעיף הקודם אלא שקוראים לפונקציה **<u>לפני שעבר</u>** מס׳ הבלוקים BLOCKS_TO_RESPOND

5. *Call function until K is decremented*

נניח 4 אנשים בקבוצה: דוד ברק שובל רוחמה. אנו מוציאים את כל האנשים בקבוצה אחד אחד עד שיישאר בקבוצה רק אחד (דוד). בנוסף לבדיקת האם האנשים נמצאים בקבוצה או לא + האם הchallenge פעיל או לא, רוצים לבדוק שהN קטן תמיד וכאשר הN נהיה שווה ל K אז שניהם קטנים בו זמנית.

6. *Require #3*

המקרה הזה יתקיים במצב הבא:
נניח 4 אנשים בקבוצה: דוד ברק שובל רוחמה. מצב של 2 מתוך 4. דוד שולח challenge לשובל והיא לא עונה אז מוציאים אותה מהקבוצה. כרגע דוד שולח challenge לרוחמה. ברק לא היה מודע להוצאה של שובל לכן רוצה לבדוק אם היא הוצאה מהקבוצה. אכן יש challenge פעיל אבל הtarget עכשיו הוא השתנה (רוחמה ולא שובל). לכן הrequire הזה הכרחי.

requestPayment

1. *Require #1*

המשתמש לא שייך לקבוצה.

2. *Require #2*

המשתמש מבקש לבצע העברה בסכום שאינו חיובי ממש.

3. Function Test

שליחת request ובדיקה שה-ledger מעודכן בבקשה זאת. שליחת בקשה נוספת באותו הסכום ע"י משתמש אחר ובדיקה ששתי הבקשות נמצאות ב-ledger.

approvePayment

1. *Require #1*

מי שקורא לפונקציה אינו שייך לקבוצה - נזרקת שגיאה

2. *Require #2.1*

מספר הטרנזקציה ששולחים לא תקין (0 ומטה)

3. *Require #2.2*

מספר הטרנזקציה ששולחים תקין אך לא קיים בledger (לא עשינו requestPayment)

4. *Require #3*

כאשר מנסים לבצע תשלום שמשאיר בחשבון סכום נמוך יותר כאשר יש challenge פועל

5. *Error when sending too much (without challenge)*

אם רוצים לשלוח יותר ממה שיש בחשבון לבדוק האם זורק שגיאה אוטומטית או מעביר את מה שיש

6. *Normal behavior test*

קבוצה של 2 מתוך 3, אחד עושה requestPayment, השני קורא לapprovePayment ורק אז הטרנזקציה אכן מתבצעת (בדיקת פרמטרים וכו׳)

7. *Only one person in group*

כיוון שיש רק בן אדם אחד בקבוצה, אנו רוצים שכאשר הבן אדם יצור את הבקשה (כלומר יקרא לפונקציה requestPayment) אזי הפונקציה תועבר ישר (אין צורך בקריאה לפונקציה approvePayment)

<u>דברים נוספים שחשבנו עליהם לגבי הוצאת משתמש מהקבוצה:</u>

1. נניח ויש לנו מצב של 3 מתוך 3: שובל, דוד וברק. דוד וברק מבצעים תשלום של כל הסכום בחשבון לחשבון חדש ושניהם מאשרים. אח"כ דוד שולח challenge לברק והוא בכוונה לא עונה. נגיע למצב של 2 מתוך 2. אם במצב זה היינו מאשרים טרנזקציות עם 2 מתוך 2 אזי הטרנזקציה הייתה מאושרת ודוד וברק היו "מרמים" את שובל. לכן- לא נרצה לתת אפשרות לאשר טרנזקציה יותר מפעם אחת עבור משתמש מסוים.
2. כתוצאה מסעיף 1 נובע שאם למשל יש לנו שלושה אנשים בחשבון. שניים אישרו והשלישי עדיין לא ואז הוא הוצא מהקבוצה, שני האנשים שנשארו יצטרכו לבצע requestPayment מחדש ולאשר מחדש כיוון שהטרנזקציה הקודמת "הלכה לאיבוד".
3. כאשר יש challenge פעיל אנו רוצים לתת אפשרות לבקש ולאשר טרנזקציות. אנו רוצים זאת כיוון שלא רוצים לתת למשתמש בקבוצה אפשרות לחסום טרנזקציות מלקרות. אם כאשר יש challenge פעיל לא היה ניתן לבצע תשלומים אזי משתמש מסוים היה "סופג את הקנס" שבשליחת challenge ומונע מטרנזקציות להתרחש.