תרגיל בית 4

ניתן להגיש עד 20.01.2019 עד השעה 23:59

בתרגיל זה עליכם לכתוב תוכנית בשפת C שרצה תחת UNIX כפתרון לתרגיל.
יש להגיש 3 קבצים:

1. קובץ hw4.c עם קוד התוכנית מתועד.
   קוד התכנית נדרש להיות נכון, מסודר, מקבילי ככל האפשר וללא שכפול קוד מיותר.
2. קובץ הרצה hw4.out
3. קובץ WORD עם שמות הסטודנטים וצילום קבצי הרצה (פרטים בהמשך) ומסך הרצה.

המשימה:

בתרגיל בית זה נממש בנק בו מוחזקים חשבונות ומספר משתנה של כספומטים דרכם ניתן לגשת לחשבונות אלו. על הפתרון להיות מקבילי ככל האפשר.

עליכם לממש:

1. N כספומטים מקומיים (ATMs). לכל ATM מספר סידורי.
2. בנק – בבנק שמורים כל החשבונות (עליכם להחליט כיצד)
3. חשבון - לכל חשבון:
   - מספר חשבון
   - סיסמא
   - יתרה

התכנית מתחילה באתחול מספר הכספומטים המתקבל כקלט מהמשתמש....

הגדרות – דרישות למימוש:

פעולה – קיימות מספר פעולות: פתיחת חשבון, משיכה, בירור יתרה, הפקד והעברה בין חשבונות. כל פעולה לוקחת שניה, כלומר בעת ביצוע כל פעולה יש לבצע sleep(1) **(אם יש צורך במנעול, ה sleep(1) יימצא בתוכו ז"א לא משחררים מנעול שדרוש לפעולה למשך שנייה)** ורק אז לסיים ולשחרר את החשבון.

חשבון – יש לשמור על נכונות לדוגמא: עדכון החשבון מ-2 כספומטים שונים בו זמנית זה לא טוב. **יש לממש קוראים כותבים בתרגיל זה על החשבונות! מימוש זה ייבדק!**

כספומט - מתעורר פעם ב-100 מילישניות ומבצע פעולה בודדת ואז שוב נרדם לעוד 100 מילישניות. הפעולות שעל הכספומט לבצע ניתנות לו בקובץ txt, שם הקובץ מתקבל כקלט מהמשתמש. *

---

*ניתן להשתמש בפונקציה usleep הנמצאת ב unistd.h.

הבנק – כל 3 שניות יש להדפיס למסך את מצב החשבונות בבנק. יכול להיות שחשבון מסויים נעול וההדפסה תתעכב וזה בסדר.

יש לממש את התכנית בצורה הבאה:

עליכם לכתוב תוכנית המממשת את הבנק ואת החשבונות תוך שימוש בספריית pthreads של לינוקס.

הנחיות למימוש:

1. על התוכנית לשמור לקובץ log.txt הודעה על כל אירוע שמתרחש בבנק.
2. כל כספומט מקומי מקבל קובץ קלט משלו לדוגמא עבור כספומט N - ATM_n_input_file.txt , שם הקובץ מתקבל כקלט בהרצת התוכנית.
3. טיפ: נא לממש את המנגנון המטפל בבקשות של הכספומטים בעזרת פונקציה המקבלת שורת טקסט המכילה את אחת הפקודות שרשומות בסעיף 4 למטה.
4. הפעולות הנתמכות ע"י הכספומטים:

- פתיחת חשבון:

```
O <account> <password> <initial_amount>
```

אם קיים חשבון בעל אותו המספר תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – account with the same id exists
```

אם החשבון נפתח בהצלחה תחזור ללוג ההודעה:

```
<ATM ID>: New account id is <id> with password <pass> and initial balance <bal>
```

- הפקדה:

```
D <account> <password> <amount>
```

אם הסיסמא שגויה תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect
```

אם ההפקדה התבצעה בהצלחה תחזור ללוג ההודעה:

```
<ATM ID>: Account <id> new balance is <bal> after <amount> $ was deposited
```

- משיכה:

```
W <account> <password> <amount>
```

אם הסיסמא שגויה תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect
```

אם ערך המשיכה גדול מהיתרה תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – account id <id> balance is lower than <amount>
```

אם המשיכה התבצעה בהצלחה תחזור ללוג ההודעה:

```
<ATM ID>: Account <id> new balance is <bal> after <amount> $ was withdrew
```

- בירור יתרה:

```
B <account> <password>
```

אם הסיסמא שגויה תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect
```

אם הבירור התבצע בהצלחה תחזור ללוג ההודעה:

```
<ATM ID>: Account <id> balance is <bal>
```

- סגירת חשבון:

```
Q <account> <password>
```

אם הסיסמא שגויה תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect
```

אחרת החשבון ייסגר (יימחק) ויודפס ללוג:

```
<ATM ID>: Account <id> is now closed. Balance was <bal>
```

- העברה מחשבון לחשבון:

```
T <account> <password> <target_account> <amount>
```

לאחר ביצוע ההעברה יתרת חשבון היעד תגדל ב`amount` ויתרת חשבון המקור תקטן ב`amount`.

אם הסיסמא שגויה תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect
```

אם ערך ההעברה גדול מיתרת חשבון המקור תחזור ללוג הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – account id <id> balance is lower than <amount>
```

אם ההעברה התבצעה בהצלחה תחזור ללוג ההודעה:

```
<ATM ID>: Transfer <amount> from account <account> to account <target_account> new account
balance is <account_bal> new target account balance is <target_bal>
```

בכל הפעולות אם יש ניסיון גישה לחשבון שלא קיים (או שנסגר) תודפס הודעת שגיאה:

```
Error <ATM ID>: Your transaction failed – account id <id> does not exist
```

במקרה של העברה מחשבון לחשבון יש לבדוק ראשית את החשבון המקור ולאחר מכן את חשבון היעד ולהדפיס רק הודעת שגיאה אחת (כלומר אם גם המקור וגם היעד לא קיימים יש להדפיס שגיאה רק עבור המקור)

5. על התכנית להדפיס את מצב כל החשבונות בסדר עולה עפ"י מספר חשבון כל 3 שניות. לדוגמא:

```
Current Bank Status
Account 123: Balance – 12  $ , Account Password – 1234
Account 124: Balance – 100 $ , Account Password – 0000
Account 133: Balance – 10  $ , Account Password – 5555
.
.
The Bank has 1500 $
```

לשים לב (דגשים):

הבנק פועל כל עוד יש פקודות בקבצי הקלט שלא התבצעו.

הרעיון העיקרי בתרגיל הינו יצירת חוטים וסינכרון ביניהם כך שתתקבל מקביליות. לכן, בכל פעולה שאתם מממשים חישבו אם דרוש סינכרון, ואם כן כיצד לבצע אותו בצורה הטובה ביותר.

ניקוד מלא יינתן לתרגיל בו תהיה **מקבילות מירבית – חובה לממש מנגנון קוראים כותבים עבור החשבונות של הבנק (כולל מנגנון קוראים/כותבים על המבנה של מאגר הנתונים אם צריך נעילות גם שם)**.

- כל ישות בתוכנית (בנק, כספומטים...) צריכה להיות עצמאית (רמז: thread לכל אחד).
- על כל ישות (כספומט/בנק) להפריע כמה שפחות לישויות האחרות, כלומר, בכל פעולה שנעשית תבצעו את ההגנה המינימלית הדרושה על מבני הנתונים. לדוגמא, כאשר כספומט A מבצע משיכה מחשבון n לכספומט B אסור לבצע פעולות באותו החשבון. הפתרון הטרוויאלי יהיה ל"נעול" את כל החשבונות כאשר מתבצעת פעולה, כך שאף פעולה אחרת לא תוכל להתבצע. הגנה כזו כמובן אינה מינימלית **ואינה** מביא למקביליות מירבית.

הנחות:

נניח כי אי אפשר להיכנס למינוס, כלומר היתרה בכל החשבונות תהיה חיובית, אם יש ניסיון לבצע משיכה גדולה מהיתרה המשיכה תיכשל.

מספר חשבון והיתרה הינם בגבולות הint, הסיסמא הינה מספר בן ארבע ספרות.

ניתן להניח כי הקלט הנקלט מהקבצים הינו קלט חוקי ("חוקי הפורמט") אך הפעולות לא בהכרח, לדוגמא ניסיון משיכה עם סיסמא שגויה – הפעולה תיכשל.

דוגמא לקובץ קלט:

```
O 12345 0000 100
W 12345 0000 50
D 12345 0000 12
O 12346 1234 45
```

קימפול התוכנית:

gcc – l pthread source-file –o exec-file

כלומר:

gcc – l pthread hw4.c –o hw4.out

הרצת התכנית תכלול את שמות קבצי הקלט בשורת הפקודה !!!

לדוגמא ההרצה:

./hw4.out ATM1.txt ATM2.txt ATM3.txt

יוצרת 3 כספומטים בבנק שלכם (בתכנית) ומניחה שיש 3 קבצי txt עפ"י השמות הנתונים כך ש ATM1.txt הוא קובץ קלט לכספומט 1, ATM2.txt הוא קובץ קלט לכספומט 2 וכך הלאה. (דוגמא לקובץ קלט בראש עמוד זה.)
בנוסף, במהלך הרצת התכנית ייווצר קובץ בשם **log.txt** המכיל תיעוד לכל אירוע שהתרחש בבנק (כפי שמתואר למעלה). קובץ זה ייבדק!

יש להגיש קובץ WORD המכיל:

1. שמות 2 המגישים (שמות ות.ז).
2. קוד התכנית שלכם.
3. פירוט קבצי הקלט שלכם (העתיקו את התכן לקובץ WORD)
4. תצלום מסך הרצה
5. פירוט קובץ log.txt (העתיקו את התכן לקובץ WORD)

לאחר שקראתם והבנתם הכל, **אנא קראו שוב**.
אם לאחר מכן יש שאלות, נא לפנות למתרגלת.

בהצלחה ☺