

תרגיל רטוב #2



בנייה ממשק בנק מקובלי

הקדמה

שיםו לב – **מקורות הקוד תבדק. העתקת שיעורי בית היא עבירה ממשמע בטכניון על כל המשתמש מכך.**

בתרגיל זה נमש בנק בו מוחזקים חשבונות ומספר כספרתיים דרכם יהיה ניתן לבצע פעולות על החשבונות (משיכה, הפקדה, העברת בין חשבונות ועוד). הספרתיים יבצעו את הפעולות בצורה מקבילית, כאשר מטרת התרגיל היא כתיבת תוכנית מקבילתית עם חוטים באמצעות `std::thread`. בתרגיל אין להשתמש באף ספרייה אחרת הנוגעת לתוכנות מקבילי/`/`או מנגנוני סנסרין (לדוגמא – `<semaphore>`, `<mutex>` – ב-`C++` ו-`<thread>`).

כמה טיפים:

1. לפני ששאליהם שאלות בפורים – עשו מאמץ לוודא שהתשובה לשאלתכם לא כתובה בהנחיות התרגיל/נשאה כבר בפורים/בקבוצה של הקורס. דגש מיוחד לתרגיל זה – שאלות כמו "אם זה בסדר שנעלתי את X בזמן ש-...". אין חלק ממימוש הנדרש בתרגיל.
2. בחלק הבא של המפרק תמצאו תיאור כללי של התרגיל, אשר לאחריו יופיעו ההנחיות בצורה מדויקת. קראו קודם כל את התיאור הכללי על מנת להבין את המבנה של התוכנית, ולאחר מכן את ההנחיות המדויקות.
3. **תכנו קוד!** שרטטו תרשימים זרימה כללי של התוכנית. מה הם מבני הנתונים הנדרשים? איך נממש אוחם? הגדרו הצהרות של `classes/structs`, תכנו פונקציות שתרצו למשר ורק אח"כ תתחילו לכתוב. ברגע שמתחיכים למימוש מסוים שלא לוקח בחשבון את אחת הדרישות, קשה הרבה יותר לשנות.
4. נסו לזהות כל חלק בתרגיל כבעיה שדיברנו עליה בקורס עם פתרון ידוע, ונסו לחשב איך הבעיה משתמשות ומשפיעות זו על זו לפתרון כולל. למשל, ניתן למדל הרבה סיטואציות שתתkalו בהן בתרגיל כבעית קוראים-כותבים – כדי לכתוב מנגנון גנרי של קוראים-כותבים ולעשות בו שימוש חזזר.
5. לתרגיל אין שILD – תכנו בחוכמה איך אתם מטפלים בדרישות הבסיסיות של התרגיל כמו פרסור ארגומנטים והקמת משתנים.
6. למדדו להשתמש ב-`gdb` – דיבוג של תוכניות מקבילות יכול להיות מורכב מאוד, והדפסות למסך בדר"כ לא מספקות פתרון מספק לך. בקורס McM IT יש סדנא עם כל מה שצריך לדעת על `gdb`, ועם קצת חיפוש באינטרנט תוכלו להציג בדיקן איזה חוט מבצע איזה קטע קוד, מה מצב המשתנים שלו וכו'.

בצלחה!

תיאור התוכנית, ממשק משתמש ותהליך הרצה

מבנה נתונים:

בתוכנית יופיעו מבני הנתונים הבאים:

1. חשבון בנק – לכל חשבון יש מרספר חשבונ, סיסמה ויתרה נוכחות.
2. בנק – הבנק יכול את כל החשבונות הקיימים.
3. כספומטים (ATMs) – הכספיים יפעלו על החשבונות שבבנק.

אתחול התוכנית:

התוכנית תקבל כקלט מספר כלשהו של נתיבים לקביעי קלט ותאותל אותו מרמספר של כספומטים, כאשר כל כספומט יוכל קובץ קלט לעצמו. כל קובץ קלט יוכל רצף של פקודות שעל הכספיים לבצע, כאשר פורמט הפקודות יפורט בהמשך. לדוגמה, אם הקלט לתוכנית הוא 3 קביעים, יש לאותל 3 כספומטים, כאשר כספומט 1 קורא ומבצע פקודות מקובץ 1, כספומט 2 קורא ומבצע פקודות מקובץ 2, כספומט 3 קורא ומבצע פקודות מקובץ 3 וכך הלאה. מרמספר הקובץ יהיה המספר הסידורי של הכספי – קובץ מרמספר 1 מתייחס לכספי מספר 1 וכו'.

סיום התוכנית:

התוכנית תסתהים כאשר כל הכספיים סיימו לבצע את כל הפקודות, או בשגיאה שאו אפשר להתחושש ממנה (יפורט בהמשך).

פעולות ופעולות הכספיים:

על כל פעולה לקחת שנייה אחת (גם אם הפעולה נכשלה). כלומר, בעט ביצוע פעולה יש להשתמש ב-
(1)sleep (נិזון להזניח את הזמן בו החוט בפעולה ביחס לשנייה בו הוא ישן). כמו כן, כל כספומט מתעורר פעם ב-100 מילישניות, מבצע פעולה וחוזר לישון 100 מילישניות **נוסף**. כלומר, ציר הזמן של כספומט נראה סכמיהית כך:

t = 0: sleep for 100msec
t = 0.1sec: perform transaction and then sleep for 1 sec
t = 1.1sec = transaction finished, sleep for 100msec
t = 1.2sec = perform transaction and then sleep for 1 sec
t = 2.2sec = transaction finished, sleep for 100msec

וכך הלאה. על כל פעולה שמבצע כספומט להרשם בקובץ **log** שיפורט בהמשך.

פעולות:

יש לתמוך בפעולות הבאות בכספיים:

1. פתיחת חשבון
2. סגירת חשבון
3. הפקדה לחשבון
4. משיכה מחשבון
5. בירור יתרה בחשבון
6. העברת כסף בין שני חשבון
7. סגירת כספומט
8. שחזור של הבנק לסטטוס קודם

9. המרת מטבע
10. השקעה בשוק ההון

פעילות הבנק:

הבנק גובה عمלה תקופתית עם שהוא רנדומלי מכל החשבונות שקיימים בו. פעם ב-3 שניות על הבנק לגבוט מכל החשבונות %5-%1 מהיתריה הנוכחית של כל חשבון בכל מטבע אותו החשבון מחזיק. על הבנק להחזיק את חשבון עצמו **אליו אין גישה דרך כספומטם** ובו ישמר הכספי הנגבה ע"י העמלות. הבנק גם מדפיס למשך את מצב כל החשבונות בו בצורה תקופתית.

קריאה לתוכנית:

התוכנית תקרא בצורה הבאה מה-command line:

```
./bank <number of VIP threads> <ATM input file 1> <ATM input file 2> ...
```

המשתנה `number of VIP threads` יפורט בהמשך בחלק של פקודות VIP. ניתן להניח כי `VIP threads` הוא מספר שלם גדול/שווה ל-0 וכי כל הנתיבים לקבצים תקינים, אין צורך לבצע validation.

פירוט מימוש

מימוש פקודות:

יש למשם את הפקודות כפי שמתוארות למטה. על כל פקודה לייצר הדפסה מתאימה לקובץ לוג בהתאם לסוג הפקודה ולהנחות שלמטה. פורמט הפקודה הוא הצורה שבה הפקודה מופיעה בקבצי פקודות של כל כספומט. כל אובייקט שמופיע בסוגרים מושלים כך:

```
<object>
```

מתאר משתנה כלשהו שיש להחליף בערך המתאים (בздפסות, בפקודות וכו'). אין להדפיס את הסוגרים המושלים.

קובץ לוג:

עליכם ליצור את קובץ הלוג בעצמכם ועל שמו להיות `txt.log`, בthicquia בה נמצא קובץ ההרצה של הבנק. ניתן להשתמש בקריאות המערכת המובנות בLINQSL לטיפול בקבצים (`open`, `close`, `write` וכו') או במחלקות מובנות ב-`C++` המאפשרות לטיפול בקבצים כמו `ofstream`. על הכתיבה לקובץ להיות תקינה – כמובן, יש למשם מנגן כתובים על הקובץ על מנת לוודא שכל חוט יכתוב את כל השורה שלו בצורה מלאה וריך ואז יכנס חוט אחר.

הערה היסטורית:

הימים הם ימי אמצע (מאי) 2002, ולכן מפתח המקרה הוא 1 דולר שווה ל-5 שקלים (הנתון היה 4.9 ניגל לנוחות), מפתחינו אנחנו יודעים.

תיאור פקודות:**1. פתיחת חשבון: O <account> <password> <initial amount ILS> <initial amount UDS>**

- א. אם קיים חשבון בעל אותו המספר יש לכתוב לוג הودעת שגיאה:

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

- ב. אם החשבון נפתח בהצלחה יש לכתוב לוג:

<ATM ID>: New account id is <id> with password <password> and initial balance <balance ILS> ILS and <balance USD> USD

2. הפקודה: D <account> <password> <amount> <currency>

- א. אם הסיסמה שגoya יש לכתוב לוג הודעת שגיאה:

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

- ב. אם הפקודה הבצעה בהצלחה יש לכתוב לוג:

<ATM ID>: Account <id> new balance is <balance ILS> ILS and <balance USD> USD after <amount> <currency> was deposited

3. משיכה: W <account> <password> <amount> <currency>

- א. אם הסיסמה שגoya יש לכתוב לוג הודעת שגיאה:

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

- ב. אם ערך המשיכה גדול מהיתריה יש לכתוב לוג הודעת שגיאה:

Error <ATM ID>: Your transaction failed – account id <id> balance is <balance ILS> ILS and <balance USD> USD is lower than <amount> <currency>

- ג. אם המשיכה הבצעה בהצלחה יש לכתוב לוג:

<ATM ID>: Account <id> new balance is <balance ILS> ILS and <balance USD> USD after <amount> <currency> was withdrawn

4. בירור יתרה: B <account> <password>

- א. אם הסיסמה שגoya יש לכתוב לוג הודעת שגיאה:

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

- ב. אם בירור יתרה הבצע בהצלחה יש לכתוב לוג:

<ATM ID>: Account <id> balance is <balance ILS> ILS and <balance USD> USD

5. סגירת חשבון: Q <account> <password>

- א. אם הסיסמה שגoya יש לכתוב לוג הודעת שגיאה:

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

- ב. אם הסיסמה נכונה החשבון ימחק מהבנק ויש לכתוב לוג:

<ATM ID>: Account <id> is now closed. Balance was <balance ILS> ILS and <balance USD> USD

6. העברה בין חשבונות: T <source account> <password> <target account> <amount> <currency>

- א. לאחר ביצוע העברת יתרה בחשבון המקור תקען ב-amount ובהתאםו תנדל היתרה בחשבון היעד.

ב. ניתן להניח שהחשבון המקורי והחשבון היעד לא יהיו זהים.

- ג. אם הסיסמה שגoya יש לכתוב לוג הודעת שגיאה:

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

- ד. אם ערך המשיכה גדול מהיתריה יש לכתוב לוג הודעת שגיאה:

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

- ה. אם העברת יתרה הבצעה בהצלחה יש לכתוב לוג:

<ATM ID>: Transfer <amount> <currency> from account <source account> to account <target account> new account balance is <source balance ILS> ILS and <source balance USD> USD new target account balance is <target balance ILS> ILS and <target balance USD> USD

7. סגירת כספומט: C <target ATM ID>

- א. כל כספומט יכול להורות לכל כספומט אחר לסיים את ריצתו.

ב. הכספיום יבקש מהבנק לסגור את הכספיום המזוהה עם ID ATM. בכל הדפסה של מצב החשבונות (יפורט בהמשך), הבנק יבודק אם יש לו בקשות לסגירת כספיומים, ובמידה וכן יבצע אותן.

ג. כספיום שהבנק הורה לו לסייע את ריצתו יסיים את הפוקודה שהוא מבצע כרגע ועוד ישחרר את כל המושבים שלו ולא יבצע פקודות נוספות.
ד. אם ID ATM אינו מזוהה כספיום (כלומר אם argc > ID ATM) יש לכתוב ללוג הודעה שגיאה:
Error <source ATM ID>: Your transaction failed – ATM ID <ATM ID> does not exist

ה. אם הסגירה המבצעת בהצלחה, על הבנק לכתוב לקובץ הלוג:
Bank: ATM <source ATM ID> closed <target ATM ID> successfully

ו. אם הסגירה נכשלה בעקבות סגירת Atm שהינו סגור כבר, יש לכתוב ללוג הודעה שגיאה:
Error <source ATM ID>: Your close operation failed – ATM ID <ATM ID> is already in a closed state

8. **R <iterations>**

א. על מנת לטפל במקרה של נפילות השרת, הבנק יתמוך באופציה לבצע שחזור (rollback) – ליטופוס קודם שלו. כפי שתראו בהמשך, על הבנק להדפיס כל חצי שנייה את הסטוס שלו – הבנק יאפשר, ע"י פקודות כספיום, להחזיר את הבנק ליטופוס המודיק שבו היה בכל אחת מההדפסות האלו.

ב. החזרת הבנק ליטופוס קודם תשחזר את ה-balance של כל החשבונות לערך שהוא, תסגורן חשבונות שלא נפתחו עדין, תפתח חשבונות שנסגרו וכו'. ניתן למשך זאת בכל דרך ששמירת את הנכונות ושומרת על מקובלויות מרבית.

ג. על הבנק לשמר את המצב האחרון שלו כדי אחוריה – מכיוון שמתבצעת הדפסת סטוס כל חצי שנייה, יש "לזכור" כ-120 סטטוסים של הבנק אחוריה.

ד. ניתן להניח שלא תתקבל בקשה לשחזר של יותר מ-120 סטטוסים (כלומר, הארגומנט R קטן מ-121). ניתן גם להניח כי R יהיה מספר שלם וחובי, כלומר $120 \leq R < 0$.

ה. הפקטור iterations יספר אחוריה ליטופוס הרצוי – למשל, 1 R יחזיר את הבנק ליטופוס אחד אחריה, ו-32 R יחזיר את הבנק 32 סטטוסים אחריה.

ו. לאחר שהבוצע השחזר יש לכתוב ללוג:

<ATM ID>: Rollback to <iterations> bank iterations ago was completed successfully

בכל הפעולות אם יש ניסיון גישה לחשבון שלא קיים (או נסגר) יש להדפיס הודעה שגיאה ללוג:

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

כאשר במקרה של פקודה העברה מחשבון לחשבון יש לבדוק קודם שחשבון המקור קיים (ובמידה ולא קיים, לעצור את הפעולה ולהדפיס שגיאה) ולאחר מכן שחשבון היעד קיים (ובמידה ולא קיים, לעצור את הפעולה ולהחזיר שגיאה).

9. **המרת מטבע:**

X <account> <password> <source currency> to <target currency> <amount in source>

א. אם הסיסמה שגיה יש לכתוב ללוג הודעה שגיאה:

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

ב. אם ערך ההמרה גדול מהיתריה במטבע יש לכתוב ללוג הודעה שגיאה:

Error <ATM ID>: Your transaction failed – account id <id> balance is <balance ILS> ILS and <balance USD> USD is lower than <amount in source> <currency>

ג. אם ההמרה המבצעת בהצלחה יש לכתוב ללוג:

<ATM ID>: Account <id> new balance is <balance ILS> ILS and <balance USD> USD after <amount> <currency> was exchanged

10. השקעה בשוק ההון: `<account> <password> <amount> <currency> <time is sec>`

פעולה זו נוספת למערכת מАЗ שמנהל הבנק קרא את הספר "השקעות לSO-ים". אנחנו מאמינים שחושוב שבعالית החשבונות ישקיעו את כספו ולא ישאירו אותו כולו בעו"ש, כדי שיוכלו לדאוג טוב יותר לעתידם.

ההשקעה תהיה לתקופה ידועה מראש בptime של 3 אחוז לשניה (הלוואי עליינו). הfrmater time חייב להיות כפולה של 5. החישוב מתבצע לפי ריבית דרייבית.

$$\text{final amount} = \text{amount} \cdot 1.03^{\text{time}}$$

בתרגיל זה איננו ממשים את מערכת שוק ההון, ולכן הכספי שהושקע אינם גלוי לנו במהלך פעולות שונות בחשבון. אין צורך לעדכן הכספי הופקד מחדש לחשבון בזמן אמת. בסיום תקופת ההשקעה, כאשר הערך המעודכן חוזר למערכת, אנו מוצפים לראות את הסכום המעודכן הן בהדפסת מצב הבנק והן בפעולות השונות המבצעות במערכת.

א. אם הסיסמה שגוייה יש לכתוב לוג הודעת שגיאיה:

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

ב. אם ההשקעה המבצעת בהצלחה יש לכתוב לוג:

`<ATM ID>`: Account `<id>` new balance is `<balance ILS>` ILS and `<balance USD>` USD after `<amount>` `<currency>` was invested for `<time>` sec

:persistent

- לכל פקודה ניתן להוסיף את המילה "PERSISTENT", בסוף המילה בלבד עם רווח אחד אחרי שאר הפקודה.
- כשהAMILA נוכח בפקודה, החוט שמקבל את הפקודה ותחל בלבנות לבצע את הפקודה כרגע (בדוק אם היא לא נוכח). אם מילה נוכח והפקודה נכשלה (לפי השיקולים שראינו לעיל), החוט ימתין למשך זמן הפקודה ואז ינסה שוב לבצע אותה פקודה בבדיקה.
- אם הפקודה נכשלה, החוט לא ידפיס לוג את השגיאה. אם הפקודה נכשלה שוב, החוט ידפיס לוג את השגיאה וימשייר לביצוע הפקודה הבא בקובץ שלו.

:VIP

- לטיפול בלקחות מיוחדים שימושים שימושים על חשבון VIP, לכל פקודה ניתן להוסיף את המחרוזת "VIP=X", כאשר X הוא מספר בין 1 ל-100. המחרוזת תופיע בסוף המילה בלבד עם רווח אחד.
- במקרה זה החוט של הכטומט שמקבל את הפקודה יכתוב אותה למילה נחונים מיוחד של פקודות.
- הבנק יקים מספר כלשהו של חוטים מיוחדים שאחראים לטפל אך ורק בבקשת של ל��ות מיוחדים, והם יבצעו פקודות מסוימות התור ללא דיחוי (כלומר, ללא השינה שמבצעים חוטי הכספיים) ולפי סדר עדיפויות שנקבע על פי הקבוע X מגביה לנמוך (כלומר, הבקשת של ל��ות עם VIP=90 תבוצע לפני בפני בקשה של ל��ות עם VIP=10, כאשר שתיהן בתור במקביל).
- מספר החוטים שמבצעים פקודות VIP ניתן לתוכנית ארגומנט ב-line command כפי שפורט בחלק על הקרייה לתוכנית.
- על חוטים המבצעים פקודות VIP לבצע פעולה בצורה זהה לחולטן לחותים שאינם חוטי VIP, פרט לשינה שחותמים רגילים מבצעים.
- על חוטים המבצעים פקודות VIP לכתוב לוג בצורה זהה לחולטן לחותים שאינם חוטי VIP. אין צורך למכש עדיפות לחוטי-VIP בכתיבה לקובץ (זהו מנגנון שנקרא priority lock).

הערות והנחיות נוספות בנוגע לפועלות הכספיים והפקודות:

1. על כל כספיים להיות ממומש בחוט נפרד.
2. מספר חשבון הוא מספר בגבולות 700 וניתן להניחס שאינו מתחיל ב-0.
3. סימנא היא מספר בין 4 ספרות.
4. ניתן להניחס כי בפתיחת חשבון תועבר יתרה א-שלילית בשני המטבעות.
5. ניתן להניחס כי הקלט בקבצים חוקי (כלומר, אין פקודות בפורמט שונה מהופיע לעליה ואין צורך לטפל בשגיאות בפורמט הפקודות עצמן).
6. על פקודות הקשורות לאותו חשבון להופיע בלוג בסדר שבו התרבזו ע"י הכספיים – ככלומר, אם פקודה A התרבז לפניו פעולה B על חשבון 1 אז פעולה A תופיע בלוג לפני פעולה B. לעומת זאת, אין משמעות לסדר פעולות בלוג בין חשבוןות שונות – למשל, אם פעולה A התרבזה על חשבון 1 לפניו ופעולה B התרבזה על חשבון 2 (כਮון לא פעולות transfer ביןיהם), אין חשיבות לסדר הופעת הפעולות בלוג.
7. פעולה מתבצעת אך ורק במקרים הרשומים בה (אליא אם כן מערבת 2 מטבעות שונים בהגדלה) אם פעולה בוצעה על מטבע מסוים, יש להתייחס רק אליו בביטוי, לדוגמה גם אם יש לי מיליון דולר (אכן) וshall בודד בחשבון וניסיתי לבצע פעולה בונגיג לשקלים בחשבון שלא יכול להתרבז (הגדלה). אין הטערכות כלל בכaspd הדולרי שהחשבון שלו מחזיק.
8. כאשר חשבון בנק משקיע את כספו בשוק ההון הסכום זהה חסום ולא ניתן למשוך אותו חזרה או לבצע בו פעולות, גם אם זה אומר שפעולות יתבטלו במערכת.

דוגמאות לקובץ קלט:

O 12345 1234 100 0

W 12345 1234 50 ILS

D 12345 1234 12 ILS PERSISTENT

O 12346 0234 45 55

W 12345 0224 35 ILS

R 1

C 1

פעולות בנק:

פעם ב-3 שניות על הבנק לגבות מכל החשבונות 5%-1% מהיתריה הנוכחית של כל חשבון, על האחוז הספציפי להיות מוגREL באופן אקראי (יש לשתמש במנגנון סטנדרטיים של השפה הנבחרת לצירט מס' אקראי). עברו כל גביית עמלה יש להדפיס לקובץ הלוג:

Bank: commissions of <commission percentage> % were charged, bank gained <commission ILS> ILS and <commission USD> USD from account <account id>

על הבנק להדפיס **למספר** (לא לקובץ הלוג) את מצב כל החשבונות כל חצי שנייה את מצב כל החשבונות לפינה השמאלית העליונה של המספר, במקביל לעובודה השוטפת של הבנק. על סדר החשבונות בהדפסה להיות לפי פו, באופן הבא:

Current Bank Status

Account <id1>: Balance - <balance1 ILS> ILS <balance1 USD> USD, Account Password - <password1>

Account <id2>: Balance - <balance2 ILS> ILS <balance2 USD> USD, Account Password - <password2>

...

וכך הלאה לכל החשבונות, כאשר `2id` <1id>. בנוסף לכך, לאחר הדפסת הטעטוס ובסדר הבא:

1. הבנק יבודוק האם התקבלו בקשות לסגורת כספומטים ובמידה וכן יבצע אותן.
2. הבנק יבודוק האם התקבלו בקשות לשחזר אחרה של סטטוס הבנק ובמידה יבצע אותן. שימוש לב - המשמעות היא שיש לשמר אחרה את סטטוס הבנק וליצור אפשרות לשחזר אותן.
3. שימוש לב שהבנק קורא בלבד ולא כותב לחשבונות, אך יש ממש מנגנון קוראים-כותבים על החשבונות ולא לנעול אותם שלא לצורך.

לשימושכם פקודות `printf` מיוחדות:

- ("J2[\"033]printf - מוחקת את המספר.
- ("H1;1[\"033]printf - מדיצה את הזמן הspent לפינה השמאלית העליונה של המספר.

צורת הדפסה זאת מייצרת מעין אינימציה של מצב הבנק על המספר.

הערות בנוגע לפעולות הבנק:

1. על הדפסה לייצג תמונה מדוייקת באותו נקודה בזמן של כל החשבונות. לדוגמה, נניח שיש בבנק 100 חשבונות עם פו בין 1 ל-100, ותחלנו את הדפסה עם חשבון מספר 1 עם יתרה של \$30. ישנדפיס את חשבון מספר 100, לחשבון מספר 1 עדין צריך להיות \$30 (אחרת, תמונה המצב של הדפסה אינה מדוייקת בשום נקודה בזמן).
2. אם חשבון כלשהו נועל במהלך הדפסה, יש לעכב את הדפסה עד שניתן לקרוא מהחשבון (אחרת נוצרת הפרה להערה מספר 1).
3. הבנק פועל כל עוד ישנן פקודות בכספיים שלא בוצעו.

הערות כלליות:

1. לפניה מימוש החלק המקבילי, מומלץ לייצר שלד פשוט לתוכנית שעובד עם כספרםט יחיד ולאחר מכן להרחיב את המימוש לכספרםטים רבים.
2. על כל היבט המקבילי של התוכנית (חווטים, מנגנוני סנכרון וכו') להיות ממומש ע"י `pthreads`. למען הספר שפק, אסור להשתמש באובייקטים כגון `std::thread`, `std::mutex` והגשות שיישתמשו בהם לא יקבלו ניקוד.
3. כדאי לבדוק את התוכנית על מנת רחוב שמספר כספרםטים, מספר חוות VIP ומספר פעולות בכל קובץ, מರיצה של חוות בוודאים לעשרות אלפי חוות. מומלץ לכתוב סקריפטים בbash/פייתון על מנת לייצר קבצי קלט ולוגנים אפשריים המתאימים להם באורך שירויו.
4. הדרישת העיקרית בתרגיל היא על התוכנית להיות מקבילת כל האפשר – כלומר, אם תבצעו נעלות במקומות לא הכרחיים, יורדו עלך נקודות. חשבו בכל פעולה שאתם מימושים האם היא דרושת סנכרון מקבילי, ואם כן כיצד לבצע אותה עם חסימה מינימלית של פעולות אחרות. על כל ישوت בתוכנית להיות עצמאית ככל האפשר (רמז – ממומשת בחווט נפרד) ולהפריע לשויות האחרות כמה שפחות – לדוגמה, כאשר כספרםט 1 מבצע משיכה מחשבון כלשהו, כספרםט 2 לא יכול לקרוא אותו מחשבון וזה נעליה הכרחית. לעומת זאת, אם תנעלו את כל הבנק ביצוע של הפעולה של כספרםט 1, זה כמובן לא מביא למקבילות המורビות.
5. צורת פעולות הבנק עם יותר כספרםט אחד היא מטבעה לא דטרמיניסטיות מכיוון שתלויה בסדר זימון החווטים עליו אין התחייבות. חשבו כיצד ניתן לדבג בכל זאת (עם `gdb` עם `info threads`), כיצד ניתן להגדיר נוכנות ואיך יש לטפל בכל הדרישות השונות.
6. אין סיטואציה שבה מחשבון אמור להכנס למינוס, ובמידה וזה כן מתרחש זה מעיד על אג משמעותי – כל משיכה של סכום מעבר ליתרה הנוכחית אמורה להכשל.
7. אין להשתמש ב-`t_pthread_rwlock`.
8. יש להקפיד על הדפסות התואמות ב-100% את הדרישות בתרגיל – שנייה בהדפסות עלולה להוביל להורדת נקודות.
9. יש לכתוב ב-C או C++ בלבד.
10. הפורום עומד לשירותכם בשאלות בכל נושא הקשור לתרגיל.

טיפול בשגיאות:

אם לא הועברו פרמטרים לתוכנית או שאחד הקבצים שניתנו לא נפתח לקרואיה, יש להדפיס ל-`stderr` את השגיאה הבאה ולסייע את ריצת התוכנית:

Bank error: illegal arguments

אם קריאת מערכת נכשלת והארגומנטים שהועברו לה תקין, יש להדפיס הודעה שגיאה עם () perror ולסייע את ריצת התוכנית, בפורמט הבא:

Bank error: <syscall name> failed

הערה: כשלון של קריאת מערכת בתרגיל הרטוב הזה, בניגוד לתרגיל רטו 1, הוא סיטואציה שלא אמורה לקרות בritchת תקינה ומעידה על אג לוגי משמעותי בקוד – לדוגמה `pthread_mutex_lock` יכולה להכשל אם החווט שמנסה לנעול את המנגנון כבר מחזיק בו, מה שמעיד על צורך בשינוי הקוד ולא בטיפול כלשהו בזמן ריצה.

קומpileציה ולינקאג'

יש לוודא שהקוד מתקמפל ע"י הפקודות הבאות, בהתאם ל-C++ ו-L-C:

```
g++ -std=c++11 -g -Wall -Werror -pedantic-errors -DNDEBUG -pthread *.cpp -o bank  
gcc -std=c99 -g -Wall -Werror -pedantic-errors -DNDEBUG -pthread *.c -o bank  
הציג Werror גורם ל-gcc/warnings gcc/g++ להפוך ל-s.errors. errors. יש לוודא שהקוד שלכם מתקמפל על המcona של הקורס לא warnings/errors.
```

קוד שלא יתקמפל/יתקמפל עם אזהרות לא יכול צוין.

עליכם לספק Makefile – הכללים שחייבים להופיע בו הם:

1. כלל bank שייבנה את התוכנית
2. כלל עבור כל קובץ נפרד שקיים בפרויקט
3. כלל clean המוחק את כל תוצרי הקומPILEציה

וודאו שהתוכנית נבנתה ע"י הפקודה make. יש لكمפל וללנקג' את התוכנית עם הדגמים המופיעים לעילו. מומלץ להפריד את המימוש לקבצי .c/.cpp ו-.h. על מנת להקל על בנית התוכנית – יש להתייחס לכך בקובץ Makefile.

בנוסף לקבצים הנ"ל יש לספק קובץ readme לפי הפורמט ב下来的 תרגולי הבית. לתרגיל מצורף סקריפט הבזק בצורה חלקית את תקינות ההגשה (מבחינה טכנית, לא מבחינת התרגום עצמו). הסקריפט מצבע לשני פרמטרים – נתיב לקובץ קיז' ושם קובץ ההרצה. לדוגמה:
. /check_submission.py 123456789_987654321.zip bank

בהצלחה!