

תרגיל רטוב #2



בניית ממשק בנק מקבילי

הקדמה

שימו לב – מקוריות הקוד תבדק. העתקת שיעורי בית היא עבירת משמעת בטכניון על כל המשתמע מכך.

בתרגיל זה נממש בנק בו מוחזקים חשבונות ומספר כספומטים דרכם יהיה ניתן לבצע פעולות על החשבונות (משיכה, הפקדה, העברה בין חשבונות ועוד). הכספומטים יבצעו את הפעולות בצורה מקבילית, כאשר מטרת התרגיל היא כתיבת תוכנית מקבילית עם חוטים באמצעות pthreads. בתרגיל אין להשתמש באף ספרייה אחרת הנוגעת לתכנות מקבילי ו/או מנגנוני סנכרון (לדוגמא – `<mutex>`, `<semaphore>`, `<thread>` ב-C++ אסורות לשימוש).

כמה טיפים:

1. לפני ששואלים שאלות בפורום – עשו מאמץ לוודא שהתשובה לשאלה שלכם לא כתובה בהנחיות התרגיל/נשאלה כבר בפורום/בקבוצה של הקורס. דגש מיוחד לתרגיל הזה – שאלות כמו "האם זה בסדר שנעלתי את X בזמן ש-Y..." הן חלק ממימוש הנדרש בתרגיל.
2. בחלק הבא של המסמך תמצאו תיאור כללי של התרגיל, כאשר לאחריו יופיעו ההנחיות בצורה מדויקת. קראו קודם כל את התיאור הכללי על מנת להבין את המבנה של התוכנית, ואחר כך את ההנחיות המדויקות.
3. **תכננו קודם!** שרטטו תרשים זרימה כללי של התוכנית. מה הם מבני הנתונים הנדרשים? איך נממש אותם? הגדירו הצהרות של `classes/structs`, תכננו פונקציות שתרכזו לממש ורק אח"כ תתחילו לכתוב. ברגע שמתחייבים 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 לחשוב איך הבעיות משתלבות ומשפיעות זו על זו לפתרון כולל. למשל, ניתן למדל הרבה סיטואציות שתתקלו בהן בתרגיל כבעיית קוראים-כותבים – כדאי לכתוב מנגנון גנרי של קוראים-כותבים ולעשות בו שימוש חוזר.
5. לתרגיל אין שלד – תכננו בחוכמה איך אתם מטפלים בדרישות הבסיסיות של התרגיל כמו פרסור ארגומנטים והקמת משתנים.
6. למדו להשתמש ב-gdb – דיבוג של תוכניות מקביליות יכול להיות מורכב מאוד, והדפסות למסך בדר"כ לא מספקות פתרון מספק לכך. בקורס ממ"ת יש סדנא עם כל מה שצריך לדעת על gdb, ועם קצת חיפוש באינטרנט תוכלו להציג בדיוק איזה חוט מבצע איזה קטע קוד, מה מצב המשתנים שלו וכו'.

בהצלחה!

תיאור התוכנית, ממשק משתמש ותהליך הרצה

מבני נתונים:

בתוכנית יופיעו מבני הנתונים הבאים:

1. חשבון בנק – לכל חשבון ישמר מספר חשבון, סיסמא ויתרה נוכחית.
2. בנק – הבנק יכיל את כל החשבונות הקיימים.
3. כספומטים (ATMs) – הכספומטים יפעלו על החשבונות שבבנק.

אתחול התוכנית:

התוכנית תקבל כקלט מספר כלשהו של נתיבים לקבצי קלט ותאתחל אותו מספר של כספומטים, כאשר כל כספומט יקבל קובץ קלט פרטי לעצמו. כל קובץ קלט יכיל רצף של פקודות שעל הכספומט לבצע, כאשר פורמט הפקודות יפורט בהמשך. לדוגמא, אם הקלט לתוכנית הוא 3 קבצים, יש לאתחל 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. מספר הקובץ יהיה המספר הסידורי של הכספומט – קובץ מספר 1 מתאים לכספומט מספר 1 וכו'.

סיום התוכנית:

התוכנית תסתיים כאשר כלל הכספומטים סיימו לבצע את כל הפקודות, או בשגיאה שאי אפשר להתאושש ממנה (פירוט בהמשך).

פעולות ופעילות הכספומט:

על כל פעולה לקחת שנייה אחת (גם אם הפעולה נכשלה). כלומר, בעת ביצוע פעולה יש להשתמש ב-sleep(1) (ניתן 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 **נוספות**. כלומר, ציר הזמן של כספומט נראה סכמטית 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 בכספומטים:

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

9. המרת מטבע

10. השקעה בשוק ההון

פעילת הבנק:

הבנק גובה עמלה תקופתית עם שהיא אחוז רנדומלי מכל החשבונות שקיימים בו. פעם ב-3 שניות על הבנק לגבות מכל החשבונות 1%-5% מהיתרה הנוכחית של כל חשבון בכל מטבע אותו החשבון מחזיק. על הבנק להחזיק את חשבון משל עצמו **אליו אין גישה דרך הכספומטים** ובו ישמר הכסף הנגבה ע"י העמלות. הבנק גם מדפיס למסך 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 input validation.

פירוט מימוש

מימוש פקודות:

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

<object>

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

קובץ לוג:

עליכם לייצר את קובץ הלוג בעצמכם ועל שמו להיות log.txt, בתיקייה בה נמצא קובץ ההרצה של הבנק. ניתן להשתמש בקריאות המערכת המובנות בלינוקס לטיפול בקבצים (open, close, write וכו') או במחלקות מובנות ב-C++ המקלות על הטיפול בקבצים כמו ofstream. על הכתיבה לקובץ להיות תקינה – כלומר, יש לממש מנגנון כותבים על הקובץ על מנת לוודא שכל חוט יכתוב את כל השורה שלו בצורה מלאה ורק ואז יכנס חוט אחר.

הערה היסטורית:

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

תיאור פקודות:

1. **פתיחת חשבון:** `O <account> <password> <initial amount ILS> <initial amount USD>`
 - א. אם קיים חשבון בעל אותו המספר יש לכתוב ללוג הודעת שגיאה:
`Error <ATM ID>: Your transaction failed – account with the same id exists`
 - ב. אם החשבון נפתח בהצלחה יש לכתוב ללוג:
`<ATM ID>: New account id is <id> with password <password> and initial balance <balance ILS> ILS and <balance USD> USD`
2. **הפקדה:** `D <account> <password> <amount> <currency>`
 - א. אם הסיסמא שגויה יש לכתוב ללוג הודעת שגיאה:
`Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect`
 - ב. אם ההפקדה התבצעה בהצלחה יש לכתוב ללוג:
`<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 הסיסמא שגויה יש לכתוב ללוג הודעת שגיאה:
`Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect`
 - ב. אם ערך המשיכה גדול מהיתרה יש לכתוב ללוג 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 התבצעה בהצלחה יש לכתוב ללוג:
`<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 הסיסמא שגויה יש לכתוב ללוג הודעת שגיאה:
`Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect`
 - ב. אם בירור היתרה התבצע בהצלחה יש לכתוב ללוג:
`<ATM ID>: Account <id> balance is <balance ILS> ILS and <balance USD> USD`
5. **סגירת חשבון:** `Q <account> <password>`
 - א. אם הסיסמא שגויה יש לכתוב ללוג הודעת שגיאה:
`Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect`
 - ב. אם הסיסמא נכונה החשבון ימחק מהבנק ויש לכתוב ללוג:
`<ATM ID>: Account <id> is now closed. Balance was <balance ILS> ILS and <balance USD> USD`
6. **העברה בין חשבונות:**
 - א. לאחר ביצע ההעברה היתרה בחשבון המקור תקטן ב-amount ובהתאמה תגדל היתרה בחשבון היעד.
 - ב. ניתן להניח שחשבון המקור וחשבון היעד לא יהיו זהים.
 - ג. אם הסיסמא שגויה יש לכתוב ללוג הודעת שגיאה:
`Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect`
 - ד. אם ערך המשיכה גדול מהיתרה יש לכתוב ללוג הודעת שגיאה:
`Error <ATM ID>: Your transaction failed – balance of account id <id> is lower than <amount> <currency>`
 - ה. אם ההעברה התבצעה בהצלחה יש לכתוב ללוג:
`<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 יוכל להורות לכל כספומט אחר לסיים את ריצתו.

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

ג. כספומט שהבנק הורה לו לסיים את ריצתו יסיים את הפקודה שהוא מבצע כרגע ואז ישחרר את כל המשאבים שלו ולא יבצע פקודות נוספות.

ד. אם ATM ID אינו מזוהה כספומט (כלומר אם $ATM ID > argc$) יש לכתוב ללוג הודעת שגיאה:
Error <source ATM ID>: Your transaction failed – ATM ID <ATM ID> does not exist

ה. אם הסגירה התבצעה 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 ההודעת שגיאה:
Error <source ATM ID>: Your close operation failed – ATM ID <ATM ID> is already in a closed state

8. שחזור: <iterations> R

א. על מנת לטפל במקרים של נפילות השרת, הבנק יתמוך באופציה לבצע שחזור (rollback) לסטטוס קודם שלו. כפי שתראו בהמשך, על הבנק להדפיס כל חצי שנייה את הסטטוס שלו – הבנק יאפשר, ע"י פקודת כספומט, להחזיר את הבנק לסטטוס המדויק שבו היה בכל אחת מההדפסות האלו.

ב. החזרת הבנק לסטטוס קודם תשחזר את ה-balance של כל החשבונות לערך שהיו, תסגור חשבונות שלא נפתחו עדיין, תפתח חשבונות שנסגרו וכו'. ניתן לממש זאת בכל דרך שמשמרת את הנכונות ושומרת על מקביליות מרבית.

ג. על הבנק לשמור את המצב האחורי שלו כדקה אחורה – מכיוון שמתבצעת הדפסת סטטוס כל חצי שנייה, יש "לזכור" כ-120 סטטוסים של הבנק אחורה.

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

ה. הפרמטר iterations יספור אחורה לסטטוס הרצוי – למשל, R 1 יחזיר את הבנק סטטוס אחד אחורה, ו-R 32 יחזיר את הבנק 32 סטטוסים אחורה.
ו. לאחר שהתבצע השחזור 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 הודעת שגיאה ללוג:

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

כאשר במקרה של פקודת 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 הסיסמא שגויה יש לכתוב ללוג הודעת שגיאה:
Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect

ב. אם ערך ההמרה גדול מהיתרה במטבע יש לכתוב ללוג הודעת 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 התבצעה 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: $\langle \text{time is sec} \rangle \langle \text{currency} \rangle \langle \text{amount} \rangle \langle \text{password} \rangle \langle \text{account} \rangle$ |

פעולה זו נוספה למערכת מאז שמנהל הבנק קרא את הספר "השקעות ל-OS-ים". אנחנו מאמינים שחשוב שבעלי החשבונות ישקיעו את כספם ולא ישאירו את כולו בעו"ש, כדי שיוכלו לדאוג טוב יותר לעתידם.

ההשקעה תהיה לתקופה ידועה מראש בריבית של 3 אחוז לשנתי (הלוואי עלינו). הפרמטר time חייב להיות כפולה של 5. החישוב מתבצע לפי ריבית דריבית.

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

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

- א. אם הסיסמא שגויה יש לכתוב ללוג הודעת שגיאה:
Error <ATM ID>: Your transaction failed – password for account id <id> is incorrect
- ב. אם ההשקעה התבצעה בהצלחה יש לכתוב ללוג:
<ATM ID>: Account <id> new balance is <balance ILS> ILS and <balance USD> USD after <amount> <currency> was invested for <time> sec

פקודות persistent:

1. לכל פקודה ניתן להוסיף את המילה "PERSISTENT", בסוף המילה בלבד עם רווח אחד אחרי שאר הפקודה.
2. כשהמילה נוכחת בפקודה, החוט שמקבל את הפקודה יתחיל בלנסות לבצע את הפקודה כרגיל (בדיוק כמו אם היא לא נוכחת). אם מילה נוכחת והפקודה נכשלת (לפי השיקולים שראינו למעלה), החוט ימתין למשך שאר זמן הפקודה ואז ינסה שוב לבצע את אותה פקודה בדיוק.
3. אם הפקודה נכשלה, החוט לא ידפיס ללוג את השגיאה. אם הפקודה נכשלת שוב, החוט ידפיס ללוג את השגיאה וימשיך לביצוע הפקודה הבאה בקובץ שלו.

פקודות VIP:

1. לטיפול בלקוחות מיוחדים שמשלמים 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.
2. במקרה זה החוט של הכספומט שמקבל את הפקודה יכתוב אותה למבנה נתונים מיוחד של פקודות.
3. הבנק יקים מספר כלשהו של חוטים מיוחדים שאחראיים לטפל אך ורק בבקשות של לקוחות מיוחדים, והם יבצעו פקודות מאותו התור ללא דיחוי (כלומר, ללא השינה שמבצעים חוטי הכספומטים) ולפי סדר עדיפויות שנקבע על פי הקבוע X מגבוה לנמוך (כלומר, הבקשה של לקוח עם VIP=90 תבוצע לפני בקשה של לקוח עם VIP=10, כאשר שתיהן בתור במקביל).
4. מספר החוטים שמבצעים פקודות VIP ניתן לתוכנית כארגומנט ב-command line כפי שפורט בחלק על הקריאה לתוכנית.
5. על חוטים המבצעים פקודות VIP לבצע פעולות בצורה זהה לחלוטין לחוטים שאינם חוטי VIP, פרט לשינה שחוטים רגילים מבצעים.
6. על חוטים המבצעים פקודות VIP לכתוב ללוג בצורה זהה לחלוטין לחוטים שאינם חוטי VIP. אין צורך לממש עדיפות לחוטי ה-VIP בכתיבה לקובץ (זהו מנגנון שנקרא priority lock).

הערות והנחיות נוספות בנוגע לפעילות הכספומטים והפקודות:

1. על כל כספומט להיות ממומש בחוט נפרד.
2. מספר חשבון הוא מספר בגבולות int וניתן להניח שאינו מתחיל ב-0.
3. סיסמא היא מספר בין 4 ספרות.
4. ניתן להניח כי בפתיחת חשבון תועבר יתרה אי-שלילית בשני המטבעות.
5. ניתן להניח כי הקלט בקבצים חוקי (כלומר, אין פקודות בפורמט שונה מהמופיע למעלה ואין צורך לטפל בשגיאות בפורמט הפקודות עצמן).
6. על פקודות שקשורות לאותו חשבון להופיע בלוג בסדר שבו התבצעו ע"י הכספומטים – כלומר, אם פקודה A התבצע לפני פעולה B על חשבון 1 אז פעולה A תופיע בלוג לפני פעולה B. לעומת זאת, אין משמעות לסדר פעולות בלוג בין חשבונות שונים – למשל, אם פעולה A התבצעה על חשבון 1 לפני ופעולה B התבצעה על חשבון 2 (כמובן לא פעולות transfer ביניהם), אין חשיבות לסדר הופעת הפעולות בלוג.
7. פעולה מתבצעת אך ורק במטבע הרשום בה (אלא אם כן מערבת 2 מטבעות שונים בהגדרתה) אם פעולה בוצעה על מטבע מסוים, יש להתייחס רק אליו בביצוע, לדוגמה גם אם יש לי מיליון דולר (אמן) ושקל בודד בחשבון וניסיתי לבצע פעולה בנוגע לשקלים בחשבון שלא תוכל להתבצע (מהגדרה). אין התערבות כלל בכסף הדולרי שהחשבון שלי מחזיק.
8. כאשר חשבון בנק משקיע את כספו בשוק ההון הסכום הזה חסום ולא ניתן למשוך אותו חזרה או לבצע בו פעולות, גם אם זה אומר שפעולות יתבטלו במערכת.

דוגמא 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 1%-5% מהיתרה הנוכחית של כל חשבון. על האחוז הספציפי להיות מוגרל באופן אקראי (יש להשתמש במנגנונים סטנדרטיים של השפה הנבחרת ליצירת מספר אקראי). עבור 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 **למסך** (לא לקובץ הלוג) את מצב כל החשבונות כל חצי שנייה את מצב כל החשבונות לפינה השמאלית העליונה של המסך, במקביל לעבודה השוטפת של הבנק. על סדר החשבונות בהדפסה להיות לפי id, 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 id2 < id1. בנוסף לכך, לאחר הדפסת הסטטוס ובסדר הבא:

1. הבנק יבדוק האם התקבלו בקשות לסגירת כספומטים ובמידה וכן יבצע אותן.
2. הבנק יבדוק האם התקבלו בקשות לשחזור אחורה של סטטוס הבנק ובמידה ויבצע אותן. שימו לב - המשמעות היא שיש לשמור אחורה את סטטוס הבנק ולייצר אפשרות לשחזור אותו.
3. שימו לב שהבנק קורא בלבד ולא כותב לחשבונות, לכן יש לממש מנגנון קוראים-כותבים על החשבונות ולא לנעול אותם שלא לצורך.

לשימושכם פקודות printf מיוחדות:

- printf("\033[2J") – מוחקת את המסך.
- printf("\033[1;1H") – מזיזה את הזמן הסמן לפינה השמאלית העליונה של המסך.

צורת הדפסה זאת מייצרת מעין אנימציה של מצב הבנק על המסך.

הערות בנוגע לפעילות הבנק:

1. על ההדפסה לייצג תמונה מדויקת באותו נקודה בזמן של כל החשבונות. לדוגמא, נניח שיש בבנק 100 חשבונות עם id בין 1 ל-100, והתחלנו את ההדפסה עם חשבון מספר 1 עם יתרה של \$30. כשנדפיס את חשבון מספר 100, לחשבון מספר 1 עדיין צריך להיות \$30 (אחרת, תמונת המצב של ההדפסה אינה מדויקת בשום נקודה בזמן).
2. אם חשבון כלשהו נעול במהלך ההדפסה, יש לעכב את ההדפסה עד שניתן לקרוא מהחשבון (אחרת נוצרת הפרה להערה מספר 1).
3. הבנק פועל כל עוד ישנן פקודות בכספומטים שלא בוצעו.

הערות כלליות:

1. לפני מימוש החלק המקבילי, מומלץ לייצר שלד פשוט לתוכנית שעובד עם כספומט יחיד ולאחר מכן להרחיב את המימוש לכספומטים רבים.
2. על כל ההיבט המקבילי של התוכנית (חוטים, מנגנוני סנכרון וכו') להיות ממומש ע"י pthreads. למען הסר ספק, אסור להשתמש באובייקטים כגון std::thread, std::mutex והגשות שישמשו בהם לא יקבלו ניקוד.
1. כדאי לבדוק את התוכנית על מנעד רחב של מספר כספומטים, מספר חוטי VIP ומספר פעולות בכל קובץ, מריצה של חוטים בודדים לעשרות אלפי חוטים. מומלץ לכתוב סקריפטים בבאש/פייתון על מנת לייצר קבצי קלט ולוגים אפשריים המתאימים להם באורך שרירותי.
3. הדרישה העיקרית בתרגיל היא שעל התוכנית להיות מקבילית ככל האפשר – כלומר, אם תבצעו נעילות במקומות לא הכרחיים, יורדו על כך נקודות. חשבו בכל פעולה שאתם מממשים האם היא דורשת סנכרון מקבילי, ואם כן כיצד לבצע אותו עם חסימה מינימלית של פעולות אחרות. על כל ישות בתוכנית להיות עצמאית ככל האפשר (רמז – ממומשת בחוט נפרד) ולהפריע לישויות האחרות כמה שפחות – לדוגמא, כאשר כספומט 1 מבצע משיכה מחשבון כלשהו, כספומט 2 לא יכול לקרוא מאותו חשבון וזו נעילה הכרחית. לעומת זאת, אם תנעלו את כל הבנק בביצוע של הפעולה של כספומט 1, זה כמובן לא מביא למקביליות המירבית.
4. צורת פעילות הבנק עם יותר מכספומט אחד היא מטבעה לא דטרמיניסטית מכיוון שתלויה בסדר זימון החוטים עליו אין התחייבות. חשבו כיצד ניתן לדבג בכל זאת (gdb עם info threads), כיצד ניתן להגדיר נכונות ואיך יש לטפל בכל הדרישות השונות.
5. ניתן להשתמש במשתנים מסוג int עבור הכסף ולעגל תמיד ל-int הקרוב ביותר במקרה של שברים לא שלמים.
6. אין סיטואציה שבה חשבון אמור להכנס למינוס, ובמידה וזה כן מתרחש זה מעיד על באג משמעותי – כל משיכה של סכום מעבר ליתרה הנוכחית אמורה להכשל.
7. אין להשתמש ב-pthread_rwlock_t.
8. יש להקפיד על הדפסות התואמות ב-100% את הדרישות בתרגיל – שגיאה 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 לקריאה, יש להדפיס ל-stderr את השגיאה הבאה ולסיים את ריצת התוכנית:

Bank error: illegal arguments

אם קריאת מערכת נכשלת והארגומנטים שהועברו לה תקינים, יש להדפיס הודעת שגיאה עם perror() ולסיים את ריצת התוכנית, בפורמט הבא:

Bank error: <syscall name> failed

הערה: כשולן של קריאת מערכת בתרגיל הרטוב הזה, בניגוד לתרגיל רטוב 1, הוא סיטואציה שלא אמורה לקרות בריצה תקינה ומעידה על באג לוגי משמעותי בקוד – לדוגמא pthread_mutex_lock יכולה להכשל אם החוט שמנסה לנעול את המנעול כבר מחזיק בו, מה שמעיד על צורך בשינוי הקוד ולא בטיפול כלשהו בזמן ריצה.

קומפילציה ולינקוג'

יש לוודא שהקוד מתקמפל ע"י הפקודות הבאות, בהתאמה ל-C++ ול-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
```

הדגל Werror גורם ל-warnings רגילות בקומפיילר gcc/g++ להפוך ל-errors. יש לוודא שהקוד שלכם מתקמפל על המכונה של הקורס ללא warnings/errors.

קוד שלא יתקמפל/יתקמפל עם אזהרות לא יקבל ציון.

עליכם לספק Makefile – הכללים שחייבים להופיע בו הם:

1. כלל bank שיבנה את התוכנית bank
2. כלל עבור כל קובץ נפרד שקיים בפרוייקט
3. כלל clean המוחק את כל תוצרי הקומפילציה

וודאו שהתוכנית נבנת ע"י הפקודה make. יש לקמפל וללנקג' את התוכנית עם הדגלים המופיעים למעלה.

מומלץ להפריד את המימוש לקבצי c/cpp ו-h. על מנת להקל על בניית התוכנית – יש להתייחס לכך בקובץ ה-Makefile.

בנוסף לקבצים הנ"ל יש לספק קובץ readme לפי הפורמט בנוהל תרגילי הבית.

לתרגיל מצורף סקריפט הבודק בצורה חלקית את תקינות ההגשה (מבחינה טכנית, לא מבחינת התרגיל עצמו). הסקריפט מצבע לשני פרמטרים – נתיב לקובץ zip ושם קובץ ההרצה. לדוגמא:

```
./check_submission.py 123456789_987654321.zip bank
```

בהצלחה!