מבוא למדעי המחשב – אביב 2023 – מטלה 5

נושאים: מחרוזות, מבנים, הקצאות זיכרון, רשימות מקושרות, מצביעים גנריים.
משקל מציון התרגיל: 25%
תאריך הגשה: 05.06.2023 23:50
הגשה באיחור: ניתן להגיש באיחור של יום (בהורדת 10 נקודות – ציון מקסימלי 90), יומיים (בהורדת 20 נקודות – ציון מקסימלי 80) או שלושה ימים (בהורדת 30 נקודות – ציון מקסימלי 70). לאחר מכן לא תתאפשר ההגשה (מלבד לאיחורים מוצדקים לפי תקנון האוניברסיטה).

הנחיות כלליות

שאלות בנוגע לתרגיל יש לפרסם **באופן ציבורי** בפורום הייעודי למטלה הנמצא במודל.

בקשות להארכה מסיבות מוצדקות (מילואים, לידה, אשפוז וכו׳) יש לשלוח למייל tom.ben-dor@biu.ac.il בצירוף: שם מלא, שם משתמש במערכת ההגשה, מספר תעודת זהות ומסמכים רלוונטיים לפי הצורך.

יש להקפיד מאוד על הוראות עיצוב הקלט והפלט, בדיוק על פי הדוגמאות המצורפות. אין להוסיף או להשמיט רווחים או תווים אחרים, ואין להחליף אותיות גדולות בקטנות או להיפך. חוסר הקפדה על פרטים אלו עלול לגרור הורדה משמעותית ביותר בציון התרגיל עד כדי 0. ראו עצמכם הוזהרתם!

שימו לב שאתם עוקבים במדויק אחרי ההנחיות במסמך ה-Style Guide המפורסם באתר הקורס.

עליכם לכתוב קוד על פי הוראות התרגיל ולוודא שקיבלתם 100 בבדיקה האוטומטית הראשונית, וכן שהתרגיל מתקמפל ורץ על שרתי המחלקה (planet) **ללא שגיאות וללא אזהרות**.
תרגיל שלא עומד בסטנדרטים הבסיסיים הללו יגרור, בשל הטרחה שהוא מייצר בתהליך הבדיקה שלו הורדת נקודות משמעותית בציון שלו.

להזכירם העבודה היא אישית. "עבודה משותפת" דינה כהעתקה. העתקות נבדקות על ידי מערכת ההגשה האוטומטית, ותרגיל שהועתק יגרור בין השאר ציון 0 ופגיעה בציוני התרגול הסופיים **לכל הגורמים** השותפים בהעתקה. אתם יכולים לדון בגישות לפתרון התרגיל באופן תיאורטי, אך אין לשתף קוד בשום צורה.

בפיתוח הקוד ניתן להשתמש בכל סביבת עבודה, העיקר הוא שתדעו איך לקחת את קבצי הקוד מתוך הסביבה הזו, לבדוק אותם על שרתי האוניברסיטה, ולהגיש אותם באמצעות מערכת ההגשה.

בפיתוח הקוד אין להיעזר בכלים מונחי למידת מכונה מכל סוג, על כל קטע קוד להיכתב על ידי המגיש/ה בלבד ובאופן עצמאי.

דוגמת הרצה של התרגיל נמצאת בסוף המסמך.

הקדמה

בתרגיל זה תכתבו תוכנית המדמה בנק.
בנק שומר אצלו לקוחות ועסקאות שהתרחשו בין הלקוחות. מכיוון שלבנק מצטרפים לקוחות חדשים ומתבצעות עסקאות רבות בכל דקה, הבנק בחר לממש את הרשימות שלו (רשימת הלקוחות ורשימת העסקאות) בתור רשימות מקושרות (ולא בתוך מערך).

במטלה זו תוכלו להשתמש בספריות stdio.h, stdlib.h, string.h.

להלן המבנים אשר ישמשו אתכם בפתרון התרגיל (אין להגדיר מבנה נוסף או לשנות מבנה קיים):

```
typedef struct Node {
    void *data;
    struct Node *next;
} Node;

typedef struct Account {
    unsigned int accountNumber;
    char *accountHolder;
    int balance;
} Account;

typedef struct Transaction {
    unsigned int fromAccount;
    unsigned int toAccount;
    int amount;
} Transaction;

typedef struct Bank {
    Node *accounts;
    Node *transactions;
} Bank;
```

הסבר:

- המבנה `Node` מייצג רשימה מקושרת גנרית, כך שמספיק מבנה אחד לכל סוג של רשימה מקושרת (גם רשימת החשבונות וגם רשימת העסקאות). הוא מחזיק מצביע גנרי למידע ששמור בכל תא ברשימה (במטלה זו יכול להיות מסוג חשבון או עסקה).
- המבנה `Account` מייצג חשבון בבנק. לחשבון יש מזהה ייחודי (מספר החשבון), שם בעלי החשבון (מחרוזת באורך בלתי מוגבל) ויתרה (balance).
- המבנה `Transaction` מייצג עסקה. עסקה היא תמיד העברת כסף מחשבון כלשהו לחשבון אחר.
- המבנה `Bank` שומר בתוכו את כל המידע בתרגיל – רשימת החשבונות ורשימת העסקאות.

יש לעשות שימוש בכל המבנים. כלומר, גם אם ניתן לפתור את התרגיל ללא `Bank` (פשוט לשמור שני מצביעים לשתי הרשימות) – השתמשו במבנה לצורך נראות הקוד והעבירו רק אותו לפונקציות הראשיות בכל משימה.

בתרגיל זה תשתמשו בהקצאות זיכרון דינמיות. חובה לוודא כי כל הזיכרון בו השתמשתם משוחרר בסיום התוכנית (אין דליפות זיכרון) באמצעות תוכנת `valgrind` הנמצאת על שרתי המחלקה (planet).

תפריט

בתחילת התוכנית יודפס למסך התפריט הבא:

```
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
```

לאחר מכן, על פי בחירת המשתמש, תתבצע המשימה המבוקשת.
בסיום ביצוע משימה, יש להדפיס מחדש את התפריט ולפעול על פי בחירת המשתמש.
במידה והמשתמש מקיש 0, יש לסיים את התוכנית.
במידה והמשתמש מקיש אופציה שאינה מופיעה בתפריט, יש להדפיס `Invalid option` **ולהדפיס מחדש את התפריט**.
ניתן להניח שהמשתמש יכניס תו בודד (יכול להיות כל תו) ולאחר מכן ילחץ Enter.

שימו לב: לאחר הדפסת התפריט יש לרדת שורה.

דוגמאות הרצה לתרגיל עם קלט ופלט מצופים מופיעים בסוף המסמך וכן מעודכנים במערכת הגשת התרגילים.

משימה 1 – יצירת חשבון

לאחר שהמשתמש בחר במשימה 1, הוא יתבקש לבחור מספר חשבון:

```
Enter account number:
100
```

אם המספר פנוי **וגם שונה מ-0**, המשתמש יתבקש להכניס את שם בעלי החשבון:

```
Enter account holder:
William Shakespeare
Account created successfully
```

התוכנית תוסיף לרשימת החשבונות את החשבון החדש, כאשר היתרה היא אפס.

אם המספר לא פנוי או שהוא 0, יש להדפיס:

```
Account number already exists
```

ולחזור לתפריט הראשי.

נקודות חשובות:

- ניתן להניח שיוכנס מספר בטווח `unsigned int` בתור מספר החשבון.
- שם בעלי החשבון הוא מחרוזת באורך **בלתי מוגבל**, ניתן להניח שלא ריקה ואין צורך לוודא שאכן מדובר בשם תקין.

משימה 2 – מחיקת חשבון

לאחר שהמשתמש בחר במשימה 2, הוא יתבקש להכניס מספר חשבון:

```
Enter account number:
100
```

אם החשבון נוצר על ידי המשתמש באינטראקציות קודמות, יש למחוק אותו מרשימת החשבונות ולהדפיס:

```
Account deleted successfully
```

אם החשבון לא קיים בבנק יש להדפיס:

```
Account not found
```

נקודות חשובות:

- ניתן להניח שיוכנס מספר בטווח `unsigned int` בתור מספר החשבון.
- אין צורך למחוק עסקאות בהן מעורב החשבון מרשימת העסקאות, הבנק מעוניין לשמור את העסקאות גם עבור חשבונות מחוקים.

משימה 3 – עדכון חשבון

לאחר שהמשתמש בחר במשימה 3, הוא יתבקש להכניס מספר חשבון:

```
Enter account number:
100
```

אם החשבון נוצר על ידי המשתמש באינטראקציות קודמות, יש לקבל מהמשתמש את השם החדש של בעלי החשבון ולעדכנו בהתאם:

```
Enter new account holder:
Leonardo da Vinci
```

אם החשבון לא קיים בבנק יש להדפיס:

```
Account not found
```

ולחזור לתפריט הראשי.

נקודות חשובות:

- ניתן להניח שיוכנס מספר בטווח `unsigned int` בתור מספר החשבון.
- ההנחות לגבי שם בעלי החשבון ממשימה 1 תקפים.

משימה 4 – הפקדה / משיכה

לאחר שהמשתמש בחר במשימה 4, הוא יתבקש להכניס מספר חשבון:

```
Enter account number:
100
```

אם החשבון נוצר על ידי המשתמש באינטראקציות קודמות, יש לשאול את המשתמש האם ברצונו להפקיד כסף לחשבון או למשוך כסף מהחשבון:

```
Would you like to deposit or withdraw money?
deposit
How much money would you like to deposit?
1000
Money deposited successfully; your new balance is ?
```

כאשר במקום ? תופיע היתרה המעודכנת.

אם החשבון לא קיים בבנק יש להדפיס:

```
Account not found
```

ולחזור לתפריט הראשי.

נקודות חשובות:

- ניתן להניח שיוכנס מספר בטווח `unsigned int` בתור מספר החשבון.
- כשהמשתמש מכניס את הבחירה שלו, הוא מכניס מחרוזת כלשהי באורך בלתי מוגבל. במידה והבחירה אינה **בדיוק** deposit או withdraw יש להדפיס:

```
Invalid action
```

ולחזור לתפריט הראשי.

- סכום הכסף שהמשתמש הכניס לא חייב להיות מספר, יש לבדוק אם הוכנס מספר (במידה ואכן מדובר במספר, ניתן להניח שהוא בטווח int) . אם לא הוכנס מספר או שהוא שלילי או אפס, יש להדפיס:

```
Invalid amount
```

ולחזור לתפריט הראשי.

- במידה והמשתמש ביקש למשוך יותר כסף ממה שבאפשרותו, יש להדפיס:

```
Not enough money
```

ולחזור לתפריט הראשי.

משימה 5 – ביצוע מספר עסקאות

לאחר שהמשתמש בחר במשימה 5, הוא יתבקש להכניס את ההוראות לבנק:

```
Enter instructions:
107-402:20,402-12:30,68-902:100
```

הקלט מהמשתמש הוא מחרוזת באורך בלתי מוגבל המייצגת את הפעולות שיש לבצע.
מספר הפעולות בלתי מוגבל.

כל פעולה היא שלשה – החשבון המעביר, החשבון המקבל וסכום כסף.
לדוגמה 107-402:50 היא הפעולה הבאה: חשבון מספר 107 מעביר 50 מטבעות לחשבון 402. תמיד סכום העסקה יהיה חיובי (אין עיקולים בתרגיל).
הפעולות מופרדות על ידי פסיק.

יש לבצע את הפעולות לפי הסדר בו הן הוכנסו **רק אם כולן אפשריות**.
בחזרה לדוגמה למעלה, חשבון 107 מעביר 20 מטבעות לחשבון 402 ולאחר מכן חשבון 402 מעביר 30 מטבעות לחשבון 12. אם לפני תחילת ביצוע הפעולות בחשבון 402 היו פחות מ 10 מטבעות, הפעולה השנייה לא תעבור ולכן אין לבצע את הפעולה הראשונה ובטח לא את השלישית והלאה.

אם כל הפעולות בוצעו בהצלחה, יש להדפיס:

```
Instructions executed successfully
```

נקודות חשובות:

- יש לוודא את תקינות מחרוזת ההוראות: לא מובטח דבר. אם מופיעים מספר פסיקים ברצף ניתן להתייחס אליהם כאל פסיק אחד.
- בכל מקרה של חוסר תקינות או סכום עסקה שלילי (או 0), יש להדפיס:

```
Invalid instructions
```

  ולחזור לתפריט הראשי, מבלי לבצע אף פעולה.
- מספר חשבון לא תקין – פעולה לא חוקית.

משימה 6 – צפייה בחשבון

לאחר שהמשתמש בחר במשימה 6, הוא יתבקש להכניס מספר חשבון:

```
Enter account number:
100
```

אם החשבון נוצר על ידי המשתמש באינטראקציות קודמות, יש להדפיס את מספר החשבון, שם החשבון וכל הפעולות שבוצעו בחשבון. בפורמט הבא:

```
Account #100 (Rosa Parks)
Balance: 440
Transactions:
Deposited 1000
500 to 504
40 from 68
Withdrew 100
```

אם אין עסקאות:

```
Account #100 (Rosa Parks)
Balance: 0
No transactions
```

אם החשבון לא קיים בבנק יש להדפיס:

```
Account not found
```

ולחזור לתפריט הראשי.

נקודות חשובות:

- ניתן להניח שיוכנס מספר בטווח `unsigned int` בתור מספר החשבון.
- יש לתעד במהלך התוכנית כל עסקה (משיכה, הפקדה, העברה) ברשימת העסקאות שבבנק, לפי הסדר.
- מומלץ להשתמש במספר חשבון 0 בשביל עסקאות משיכה / הפקדה, אך שימו לב שאין חשבון כזה (צפיה, עדכון, מחיקה וכו׳ עם חשבון 0 צריכים להיכשל).

הוראות הגשה

יש להגיש את התוכנית שכתבתם בקובץ בודד בשם ex_5.c לקבוצה מספר 14.

דוגמת הרצה

כאמור, דוגמה זו היא **חלקית**, עליכם לחשוב על כלל מקרי הקצה האפשריים בתוכנית ולבדוק אותם.

```
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
1
Enter account number:
100
Enter account holder:
Albert Einstein
Account created successfully
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
1
Enter account number:
200
Enter account holder:
Audrey Hepburn
Account created successfully
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
4
Enter account number:
100
Would you like to deposit or withdraw money?
deposit
How much money would you like to deposit?
1000
Money deposited successfully; your new balance is 1000
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
```

```
6. View account
4
Enter account number:
100
Would you like to deposit or withdraw money?
idk
Invalid action
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
4
Enter account number:
200
Would you like to deposit or withdraw money?
withdraw
How much money would you like to withdraw?
50
Not enough money
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
5
Enter instructions:
100-200:300,200-100:50
Instructions executed successfully
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
5
Enter instructions:
30-45:50
Invalid instructions
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
4
Enter account number:
200
Would you like to deposit or withdraw money?
withdraw
How much money would you like to withdraw?
10
Money withdrawn successfully; your new balance is 240
Choose an option:
0. Exit
```

```
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
6
Enter account number:
300
Account not found
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
6
Enter account number:
100
Account #100 (Albert Einstein)
Balance: 750
Transactions:
Deposited 1000
300 to 200
50 from 200
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
6
Enter account number:
200
Account #200 (Audrey Hepburn)
Balance: 240
Transactions:
300 from 100
50 to 100
Withdrew 10
Choose an option:
0. Exit
1. Create account
2. Delete account
3. Update account
4. Deposit / Withdraw money
5. Transferring
6. View account
0
```

נספח – המרת מחרוזת למספר

Converting a string to an integer is a common task in programming, especially when dealing with user input or file parsing. In the C programming language, there are several functions available for this purpose. In this article, we will focus on two popular functions: `atoi` and `strtol`. We will explore their usage, how they handle conversion errors, and provide examples covering all possible cases.

Converting a string to an integer using `atoi`:

The `atoi` function is a built-in function in C, which stands for "ASCII to integer." It takes a string as input and attempts to convert it to an integer value. Here's the general syntax of the `atoi` function:

```
int atoi(const char* str);
```

The `atoi` function scans the input string until it encounters a non-digit character or reaches the end of the string. It then converts the digits encountered up to that point into an integer and returns the result.

Handling Errors with `atoi`:

The `atoi` function does not provide a mechanism for reporting conversion errors. If the input string cannot be converted to an integer, it will simply return a value of 0. Therefore, it's crucial to handle error cases when using `atoi`.

Let's look at some examples to better understand the behavior of `atoi`:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    char str1[] = "123";
    char str2[] = "45abc";
    char str3[] = "abc";

    int num1 = atoi(str1);
    int num2 = atoi(str2);
    int num3 = atoi(str3);

    printf("%d\n", num1);  // Output: 123
    printf("%d\n", num2);  // Output: 45
    printf("%d\n", num3);  // Output: 0

    return 0;
}
```

In the example above, `str1` contains a valid string representation of an integer, so `atoi` successfully converts it to the integer value 123. `str2` starts with digits, but it also contains non-digit characters. `atoi` stops conversion at the first non-digit character, resulting in the value 45. Finally, `str3` does not contain any digits, so `atoi` returns 0.

Converting a string to an integer using `strtol`:

The `strtol` function provides more flexibility for converting a string to an integer in C. It allows for better error handling and also supports additional features like specifying the number base.

Here's the general syntax of the `strtol` function:

```
long strtol(const char* str, char** endptr, int base);
```

The `str` parameter is the input string that you want to convert. The `endptr` parameter is a pointer to a character pointer. After the conversion, `endptr` will be updated to point to the first character that couldn't be converted. The `base` parameter specifies the number base (e.g., 10 for decimal).

Handling Errors with `strtol`:

The `strtol` function provides a way to detect and handle conversion errors. By checking the value of `endptr` after the conversion, you can determine if the entire string was converted successfully or if an error occurred.

Here's an example that demonstrates the usage of `strtol`

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    char str1[] = "123";
    char str2[] = "45abc";
    char str3[] = "abc";

    char *endptr1;
    char *endptr2;
    char *endptr3;

    long num1 = strtol(str1, &endptr1, 10);
    long num2 = strtol(str2, &endptr2, 10);
    long num3 = strtol(str3, &endptr3, 10);

    if (*endptr1 == '\0') {
        printf("Conversion successful. Result: %ld\n", num1);
    } else {
        printf("Conversion error at position %ld\n", endptr1 - str1);
    }

    if (*endptr2 == '\0') {
        printf("Conversion successful. Result: %ld\n", num2);
    } else {
        printf("Conversion error at position %ld\n", endptr2 - str2);
    }

    if (*endptr3 == '\0') {
        printf("Conversion successful. Result: %ld\n", num3);
    } else {
        printf("Conversion error at position %ld\n", endptr3 - str3);
    }
```

```
    return 0;
}
```

In the example above, str1 is a valid string representation of an integer, so strtol successfully converts it to the long integer value 123. The endptr1 will be pointing to the null character ('\0') because the entire string was converted.

For str2, the conversion stops at the first non-digit character ('a'). The endptr2 will be pointing to 'a', indicating that an error occurred after converting the digits '45'.

Finally, str3 does not start with a digit, so strtol immediately detects the error. The endptr3 will be pointing to the beginning of the string, indicating that the conversion failed.

By examining the value of endptr after the conversion, you can identify where the error occurred in the input string.

It's important to note that strtol returns a long integer value, which provides a larger range than atoi, allowing for the conversion of larger numbers.

In conclusion, when converting a string to an integer in C, you have the choice of using atoi or strtol. While atoi is simpler to use, it lacks error reporting. On the other hand, strtol provides better error handling capabilities. By understanding their usage and behavior, you can select the appropriate function based on your specific requirements.