מטלה מספר 2 - מימוש אופרטורים עבור מטריצות ריבועיות

יושרה אקדמית

במהלך העבודה על המטלות, מותר להתייעץ עם סטודנטים אחרים ולחפש מידע באינטרנט. עם זאת, חל איסור להעתיק קטעי קוד שלמים ממקורות חיצוניים, כולל סטודנטים אחרים, אתרי אינטרנט ומודלי בינה מלאכותית (כגון ChatGPT).

יש לדווח על כל עזרה שקיבלתם, בין אם מדובר בהתייעצות עם סטודנטים אחרים או במידע שנמצא באינטרנט, בהתאם לתקנון היושר של המחלקה (https://www.ariel.ac.il/wp/cs/wp-content/uploads/sites/88/2020/08/Guidelines-for-Academic-Integrity.pdf). **במקרה של שימוש בכלי בינה מלאכותית (AI), יש לצרף את הפרומפטים שהוזנו ואת התשובות שהתקבלו**.

- **מטרת המטלה:** הבנת החומר הנלמד בהרצאות הרביעית והחמישית, כגון: העמסת אופרטורים, פונקציות חברות וכלל השלושה.
- **שימו לב!** גם במטלה זו **אסור** להשתמש בספרייה הסטנדרטית, ניתן להשתמש במערך. **זאת אומרת תצטרכו לממש מבנה/י נתונים המתאים לצרכי המטלה**. כל המיכלים הסטנדרטיים כולל vector או stack *אינם זמינים* לכם במטלה זו.
- **שימו לב!** ההגשה של המטלה ביחידים.

הוראות הגשה ב Moodle:

במערכת Moodle יש להגיש **קובץ טקסט למשל (`submission.txt`)** המכיל 3 שורות בפורמט הבא:

1. **תעודת זהות** – מספר תעודת הזהות של הסטודנט.
2. **קישור להגשה** – קישור למאגר ה-GitHub שבו נמצא הפרויקט.
3. **פרטי ה-commit האחרון** – המחרוזת המזהה של ה-commit האחרון (`commit hash`)

- דוגמה לקובץ הגשה תקין:

```
123456789
https://github.com/example-user/assignment
 e3f1c1a
```

עבר הרבה זמן מאז שסיימתם את הקורס אלגברה לינארית :-) כדי לעורר בכם נוסטלגיה, במטלה הזאת אתם תממשו מחלקה המייצגת **מטריצות ריבועיות** של מספרים ממשיים.

דרישות המטלה:

מימוש המחלקות הבאות:

הוסיפו את המחלקות במרחב שמות (namespace) חדש לבחירתכם.

מחלקה בשם `SquareMat`:

המטריצה היא מטריצה ריבועית המכילה מספרים ממשיים. עבור קלט לא תקין יש לזרוק חריגה. המחלקה תכיל את המטריצה וכן את הפונקציות הבאות:

- אופרטורי חיבור וחיסור (`mat1+mat2`,`mat2-mat1`) - מבצעים חיבור/חיסור איברים בהתאם למקומם בין שתי מטריצות בעלות אותו גודל.
- אופרטור מינוס אונארי (`-mat`) - הופך את סימן כל האיברים במטריצה (חיובי לשלילי ולהפך).
- אופרטור כפל (`mat2*mat1`) - מבצע כפל בין 2 מטריצות (https://he.wikipedia.org/wiki/%D7%9B%D7%A4%D7%9C_%D7%9E%D7%98%D7%A8%D7%99%D7%A6%D7%95%D7%AA)
- אופרטור כפל בסקלר (`scalar*matrix` וגם `matrix*scalar`)
- כפל איברי (`%`) - מכפיל כל איבר במטריצה אחת באיבר המתאים במטריצה השנייה (כלומר, האופרטור מקבל מטריצה כקלט, על המטריצה להיות באותו סדר גודל של המטריצה הנתונה).
- מודולו עם סקלר (`%`) - מבצע פעולה של מודולו על כל איבר במטריצה עם מספר שלם נתון (כלומר, האופרטור מקבל מספר שלם כקלט ומחשב את המודולו של כל איבר במטריצה עם המספר הנתון).
- חילוק בסקלר (`/`) - מחלק כל איבר במטריצה במספר סקלרי.
- אופרטור חזקה (`^`) - מעלה את המטריצה בחזקה על ידי כפל חוזר של המטריצה בעצמה.
- אופרטורי הגדלה והקטנה ב-1 (`++`, `--`) - מגדילים או מקטינים את כל איברי המטריצה באחד (ממשו גם preincrement וגם postincrement).
- אופרטור Transpose (`~`) - משחלף את המטריצה: מחליף את השורות בעמודות של המטריצה.
- אופרטור גישה לאיברים באמצעות אינדקס (`[]`) - מאפשר גישה ישירה לאיבר ספציפי במטריצה על ידי שימוש באינדקסים לדוגמה `[i][j]`. אפשרו גישה גם לצורך עדכון של איבר במטריצה.
- אופרטורי שוויון ואי-שוויון (`==`,`!=`) - במטלה הזאת מטריצות הן שוות אם סכום האיברים שלהם זהה.
- אופרטורי השוואה (`<`, `>`, `<=`, `>=`) - משווה בין שתי מטריצות. מטריצה א' גדולה ממטריצה ב' אם סכום האיברים של מטריצה א' גדול מסכום האיברים של מטריצה ב'.
- אופרטור דטרמיננטה - (`!`) - מחשב את הדטרמיננטה של המטריצה.
- אופרטורי השמה משולבים (`mat2+=mat1`, `-=`, `*=`, `/=`, `%=`) - מבצעים פעולות של חיבור, חיסור, כפל, חלוקה או מודולו על המטריצה ומעדכנים אותה ישירות (שימו לב שבמקרה של האופרטורים `%=` ו-`*=` יש לממש שני אופרטורים).
- אופרטור פלט - (`<<`) - מדפיס את המטריצה בצורה הגיונית.
- יש לבדוק את תקינות הקלט ולזרוק שגיאות במידת הצורך. בנוסף עליכם לכתוב טסטים לכל הפונקציות שכתבתם.

דרישות נוספות:

- חשוב לוודא שה-repository ציבורי.
- כתבו בתחילת **כל** קובץ את כתובת המייל שלכם.
- כתבו קוד נקי, מסודר, מחולק לקבצים, מודולרי, מתועד בצורה מספקת וכמובן בדיקות יחידה עבור כל הפונקציות.
- בדקו את תקינות הקלט ולזרוק חריגות מתאימות במידת הצורך.
- לשימושכם הקישור הבא (https://github.com/doctest/doctest doctest) בו תוכלו לראות דוגמאות נוספות לשימוש בסיפריה זו.
- יש לבדוק שאין זליגת זיכרון באמצעות `valgrind`.

- יש לצרף גם קובץ `README` עם הסבר על פרויקט, על חלוקה למחלקות וקבצים וכל מידע אחר רלוונטי.

קובץ `Makefile`:

הוסיפו לפרויקט קובץ `Makefile` הכולל את הפקודות הבאות:

- הפקודה `make Main` – להרצת קובץ ההדגמה.
- הפקודה `make test` – להרצת בדיקות היחידה.
- הפקודה `make valgrind` – בדיקת זליגת זיכרון באמצעות valgrind.
- הפקודה `make clean` - מוחקת את כל הקבצים הלא רלוונטיים לאחר ההרצה.

בהצלחה!