תאריך: **22.10.2000**

שם הקורס: **שפות תוכנה מתקדמות 1 (מועד ב')**

משך הבוחן: **שלוש שעות**

חומר עזר: **כל חומר כתוב מצולם או מודפס (כולל ספרים)**

- **<u>משקל כל שאלה 50 נק' + 10 נק' בונוס</u>**

- בכל תוכנית חייבים להיות הסברים, ושמות המשתנים חייבים להיות בעלי משמעות.
- אין להוסיף פונקציות, אופרטורים או משתנים פרט לאלו שנדרשים בשאלות באופן מפורש, **אלא אם נאמר אחרת!!!**

1. בסוף השאלה נתון **class Matrix** ובו יש שדה נתונים ב-**private** ומספר פונקציות.

ה-**default constructor** מקבל את מספר וממלא את כל המטריצה בערך זה.

constructor נוסף מקבל מטריצה ומעתיק אותה לתוך המטריצה של ה-**class**.

הפונקציה **GetElement** מחזירה איבר מתוך המטריצה.

הפונקציה **SetElement** מעדכנת איבר בתוך המטריצה עם ערך חדש.

הפונקציה **Write** כותבת את כל האובייקט לתוך קובץ בינארי לפי עמודות. **הכתיבה מתבצעת בסוף הקובץ**.

1.1. הגדירו את האופרטורים "=", "~", "!", "=!", ואת שני האופרטורים "++".

"=" – מעטיק את תוכן האוביקת לתוך אוביקט חדש.

"++" – מוסיף אחד לכל אלמנט במטריצה.

"~" מחזיר אוביקט בו השורות הפכו לעמודות.

"!" – משאירה את התוצאה באוביקט עצמו כאשר במטריצה כל איבר זה סכום כל השכנים – השכנים של (2,4) הם: (2,3), (2,5), (1,4) ו-(3,4), כלומר האיברים מלמלע, מלמטה, מימין ומשמאל (4 שכנים).

"=!" – כמו "!" רק התוצאה באוביקט חדש.

1.2. כיתבו את כל ה-**constructor** -ים שצריך (השלימו את המקומות של **TYPE??**).

1.3. כתבו את ההפונקציות **GetElement**, **SetElement** ו-**Write**. השלימו את המקומות של **TYPE??** ו-**??**.

1.4. כיתבו **operator >>** לכתיבת המטריצה לפי שורות, עם **Tab** (**\t**) בין עברי המטריצה.

- **אם יש צורך בפונקציות או אופרטורי עזר יש לנמק מדוע ולרשום אותן במקום המתאים.**
- **אם יש צורך ב-constructor-ים נוספים יש לנמק למה, ולכתוב אותם.**
- **אם יש צורך בהגדרות נוספות, הגדירו והסבירו כל הגדרה.**
- **אין להוסיף משתנים נוספים ל-class.**

```
class Matrix
{
    friend ??? operator << (???)                                    ;
  public:
    Matrix ( const TYPE?? = 0)                                      ;
    Matrix ( const TYPE?? Matrix [ 6 ] [ 6 ] )                      ;
    Matrix & SetElement ( ??? )                                     ;
    TYPE?? GetElement ( ??? )                                       ;
    ??? & Write ( const char * strFileName )                        ;
```

OPERATORS FOR OVERLOADING

```
  private:
    TYPE??          m_Matrix [ 6 ] [ 6 ]                            ;
}
```

2. נגדיר נקודה דו-מימדית – זהו class עם שני משתנים, כל משתנה מטיפוס כלשהו (שני המשתנים מאותו טיפוס). המרחק בין שתי נקודות יוגדר באמצעות: $d(\mathbf{X},\mathbf{Y}) = \left(\sum_{n=1}^{2} (x_n - y_n)(x_n - y_n)^* \right)^{1/2}$

באשר ה(*) מסמנת צמוד קומפלקסי.

עיגול מוגדר באמצעות נקודה דו-מימדיות ($\mathbf{X}$), המהווה את מרכז העיגול, ורדיוס *radius* מטיפוס כמו של אברי הנקודה.

2.1. עליכם לכתוב **class Complex** בו יש שני שדות מטיפוס **double** (ממשי ומדומה). ב-**class** זה יהיה **constructor** אשר מקבל שני מספרים,

אופרטור (+) בין שני מספרים מרוכבים,

אופרטור (=) למספרים מרוכבים,

אופרטור (*) אשר יבצע: $x * y = xy^*$, כלומר מכפלה בין מספר לצמוד שלו והתוצאה תיהיה בתוך **Complex** אחר,

אופרטור **casting** לתוך **double**. הערך המוחזר יהיה: $(double)(a+ib) = (a^2+b^2)^{0.5}$

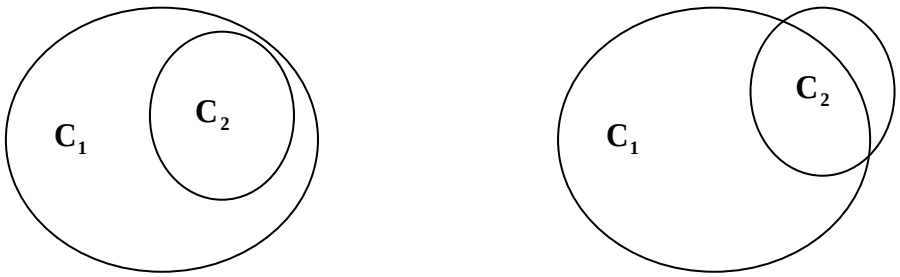
כלומר חשבו חלק ממשי בריבוע ועוד חלק מדומה בריבוע, שורש על הכל

ואופפרטור (<) אשר יבדוק האם גודלו של מספר מרוכב אחד גדול מגודלו של השני $|\mathbf{X}| < |\mathbf{Y}|$.

2.2. עליכם לכתוב **class Point**. ב-**class** זה עליכם להגדיר **constructor**, את האופרטור (-) אשר יבצע את הפונקציה $d(\mathbf{X},\mathbf{Y})$.

2.3. עליכם לכתוב **class Circle** אשר בעל שני שדות: **m_Point** מטיפוס **Point** ו- *radius*. **class** זה יכיל **constructor** ואופרטור (%). אופרטור זה בודק האם עיגול אחד מוכל בשני ומחזיר ערך בוליאני.

<u>לדוגמא:</u>



$\mathbf{C_1}\%\mathbf{C_2} = 1$
$\mathbf{C_2}\%\mathbf{C_1} = 0$

$\mathbf{C_1}\%\mathbf{C_2} = 0$
$\mathbf{C_2}\%\mathbf{C_1} = 0$

- הערה: אסור למשתנים להיות ב-**public**.
- אם יש צורך בפונקציות אופרטורים או **constructor**-ים נוספים, הוסיפו והסבירו למה.

בהצלחה