

סדנא ב- C++ – 150018

תרגיל בית מספר 3

העמסת אופרטורים

שים/י לב:

- הקפד/י על קריאות התכנית ועל עימוד (Indentation).
- הקפד/י לבצע בדיוק את הנדרש בכל שאלה.
- בכל אחת מהשאלות יש להגדיר פונקציות במידת הצורך עבור קריאות התכנית.
- יש להגיש את התרגיל על פי ההנחיות להגשת תרגילים (המופיע באתר הקורס) וביניהם:
 - השתמש/י בשמות משמעותיים עבור המשתנים.
 - יש לתעד את התכנית גם עבור פונקציות אותם הנך מגדיר/ה וכן על תנאים ולולאות וקטעי קוד מורכבים, ובנוסף, **דוגמת הרצה לכל תכנית בסוף הקובץ!**
 - הגשה בזוגות.

הערה חשובה: לתרגיל בית הזה מוגדר שבועיים להגשה, אלא אם כן קיבלת הוראה אחרת מהמרצה שלך. תיבות ההגשה הפתוחות לא מהוות היתר להגשה באיחור.

שאלה מס' 1:

שים/י לב: מתרגיל זה ואילך יש להקפיד להגדיר const במקומות המתאימים!!

חלק א:

הגדרי/י מחלקה Rational עבור המספרים הרציונליים. (אפשר להיעזר במחלקה שכתבתם בתרגיל בית 1. מה שמסומן בהגדרות הבאות **בסגול** אלו ההגדרות שזהות לתרגיל בית 1. מה שלא מסומן בכלל הן הגדרות חדשות.)

המחלקה תכלול את השדות הבאים:

- מונה (numerator)

- מכנה (denominator)

תזכורת: ערך המכנה לא יהיה 0 או שלילי

וכן את הבנאים הבאים:

- empty constructor** - מאתחל את השדות numerator ו-denominator באחד.
- constructor** – המקבל שני פרמטרים ומאתחל את השדות numerator ו-denominator לפי הפרמטרים שקבל. במקרה בו נקלט ערך 0 עבור המכנה יש להציב במכנה 1.
- שימו לב, כמו שמוגדר למעלה, אין להציב ערך שלילי במכנה יש לאתחל את השדות בהתאם.
- copy constructor**

וכן את המתודות הבאות:

- עבור כל שדה:

- מתודת הצבה (set). במכנה אין להציב 0, לכן במידה והפרמטר שמתקבל הוא 0

המתודה setDenominator(), תציב 1. שימו לב, כמו שמוגדר למעלה אין להציב

ערך שלילי במכנה. יש לבצע הצבה לשדות בהתאם.

- מתודה המחזירה את ערכו (get).

- מתודה להדפסת השבר הרציונאלי. ההדפסה תהיה בפורמט: מונה/מכנה (לדוגמא 1/2, 3/4,

54/56 וכו') עפ"י הערכים המקוריים לא לאחר צמצום.

בהדפסת המספר, אם במכנה שמור המספר 1 אז יש להדפיס רק את המונה.

- מתודת עזר לצמצום השבר

- חתימת המתודה: `void reduce()`;
- על המתודה לצמצם את העצם שקרא למתודה. כלומר עבור

```
Rational rat(2,4);
rat.reduce();
```

עכשיו המונה של rat הוא 1 ומכנה של rat הוא 2.

הערה: תצטרכו לקרוא למתודה הזו לפחות פעמיים – פעם בתוך המתודה הבודקת אם שני מספרים הם שווים, ופעם שנייה כשמדפיסים פתרון של פעולת חשבון (+, -, *, /)

חלק ב:

כדי שיתאפשר להשתמש במשתנה רציונלי באופן טבעי. עליך להוסיף למחלקה את האופרטורים: + (חיבור), - (חיסור), * (כפל), / (חילוק), ++, --, <, >, <=, >=, ==, !=

הערות:

- המתודה לחיבור (+) תהיה דומה למתודה add שכתבתם בתרגיל בית 1. על המתודה לצמצם את הסכום. כלומר עבור:

```
Rational r1(1,6);
Rational r2(1,3);
Rational r3;
r3 = r1+r2;
```

r1 נשאר 1/6

r2 נשאר 1/3

r3 הוא 1/2

- כנ"ל עבור שאר פעולות חשבוניות בינאריות (+, -, *, /)
- בפעולה של חילוק – מותר להניח שלא תתבקשו לחלק מספר עם 0. אין צורך לבדוק תקינות הקלט.
- הפעולות האונריות (+, --) לא יגרמו לצמצום השבר. כלומר עבור

```
Rational r1(2,6);
Rational r2;
r2 = r1++;
```

r1 הוא 8/6

r2 הוא 2/6

- המתודה להשוואה (אופרטור ==) תהיה דומה למתודה של equals מתרגיל בית 1, אבל הפעם המתודה תחזיר true במידה שערכם באמת שווה – כלומר עבור

- Rational r1(2,6);
- Rational r2(1,3);
- r2 == r1

הינו true

הערה -

במתודה של בדיקת השוואה תצטרכו לקרוא למתודה reduce פעמיים אחד עבור האובייקט שקרא למתודה (*this) ופעם שנייה עבור האובייקט שהתקבל כפרמטר (נקרא לאובייקט num) בגלל שאתם נדרשים להשתמש כמה שיותר ב-const, זימון הפונקציה reduce עלול לגרום לשיגאה

הדרך הקלה לעקוף את הבעיה היא לשמור את `*this` במשתנה זמני ולשמור את `num` במשתנה זמני, ולהפעיל את הפונקציה `reduce` על המשתנים הזמניים

להלן מימוש אפשרי של המתודה לפי ההנחיות הללו:

```
bool Rational::operator==(const Rational& num) const {
    Rational temp1 = *this;
    Rational temp2 = num;
    temp1.reduce();
    temp2.reduce();
    return temp1.numerator == temp2.numerator &&
        temp1.denominator == temp2.denominator;
}
```

- באותה מידה המתודה להשוואה אי שוויון תחזיר `true` במידה שערכם באמת לא שווה – כלומר עבור

- `Rational r1(2,6);`
- `Rational r2(1,3);`
- `r2 != r1`

הינו `false`

הערה -

נכתוב פתרון שלא משתמש בהרגל הרע של "שכפול קוד". בגלל שהבדיקה של אי שוויון היא היפוך של בדיקת שוויון נשתמש במתודה של בדיקת שוויון שכבר כתבנו.

אפשר להשתמש במתודה הבאה עבור אי שוויון:

```
bool Rational::operator != (const Rational& num) const
{
    return !(*this == num);
}
```

השתמש/י במחלקה שיצרת והרץ/י את התכנית ראשית המצורפת אשר תקלוט מהמשתמש שני מספרים רציונאליים (`enter two rational numbers.`) בפורמט של מונה/מכנה. על התכנית לחשב ולהדפיס את התוצאות של כל אחת מהפעולות שלהלן.

יש להשתמש ב-main המצורף עבור הרצת התכנית:

```
#include "Rational.h"
#include <iostream>
using namespace std;

enum OPERATOR {
    EXIT, ADD_PRE, ADD_POST, SUB_PRE, SUB_POST, ADD, SUB, MULT, DIV,
    GREATER, LESS_THAN, GREATER_OR_EQUAL, LESS_OR_EQUAL, EQUAL,
    NOT_EQUAL
};

int main()
{
    int numerator, denominator;
    char junk;
    int op;
    cout << "enter your choice:" << endl;
    cin >> op;
    Rational r2, r3;
    while (op != EXIT) {
        cout << "enter a rational number:" << endl;
        cin >> numerator >> junk >> denominator;
        Rational r1(numerator, denominator);
        if (op >= 5) {
            cout << "enter a rational number:" << endl;
            cin >> numerator >> junk >> denominator;
            r2.setNumerator(numerator);
            r2.setDenominator(denominator);
        }
        switch (op) {
            case ADD_PRE:
                cout << "x = ++y;" << endl;
                cout << "before operation y is: ";
                r1.print();
                r3 = ++r1;
                cout << "after operation x is: ";
                r3.print();
                cout << endl << "y is: ";
                r1.print();
                break;
            case ADD_POST:
                cout << "x = " << " y++;" << endl;
                cout << "before operation y is: ";
                r1.print();
                r3 = r1++;
                cout << "after operation x is: ";
                r3.print();
                cout << endl << "y is: ";
                r1.print();
                break;
            case SUB_PRE:
                cout << "x = --y;" << endl;
                cout << "before operation y is: ";
                r1.print();
                r3 = --r1;
                cout << "after operation x is: ";
                r3.print();
                cout << endl << "y is: ";
                r1.print();
                break;
            case SUB_POST:
                cout << "x = " << " y--;" << endl;
                cout << "before operation y is: ";
                r1.print();
```

```
        r3 = r1--;  
        cout << "after operation x is: ";  
        r3.print();  
        cout << endl << "y is: ";  
        r1.print();  
        break;  
case ADD:  
    r3 = r1 + r2;  
    r1.print();  
    cout << " + ";  
    r2.print();  
    cout << " = ";  
    r3.print();  
    break;  
case SUB:  
    r3 = r1 - r2;  
    r1.print();  
    cout << " - ";  
    r2.print();  
    cout << " = ";  
    r3.print();  
    break;  
case MULT:  
    r3 = r1 * r2;  
    r1.print();  
    cout << " * ";  
    r2.print();  
    cout << " = ";  
    r3.print();  
    break;  
case DIV:  
    r3 = r1 / r2;  
    r1.print();  
    cout << " / ";  
    r2.print();  
    cout << " = ";  
    r3.print();  
    break;  
case GREATER:  
    r1.print();  
    cout << " > ";  
    r2.print();  
    cout << " ? ";  
    if (r1 > r2)  
        cout << "yes\n";  
    else cout << "no\n";  
    break;  
case LESS_THAN:  
    r1.print();  
    cout << " < ";  
    r2.print();  
    cout << " ? ";  
    if (r1 < r2)  
        cout << "yes\n";  
    else cout << "no\n";  
    break;  
case GREATER_OR_EQUAL:  
    r1.print();  
    cout << " >= ";  
    r2.print();  
    cout << " ? ";  
    if (r1 >= r2)  
        cout << "yes\n";
```

```
        else cout << "no\n";
        break;
    case LESS_OR_EQUAL:
        r1.print();
        cout << " <= ";
        r2.print();
        cout << " ? ";
        if (r1 <= r2)
            cout << "yes\n";
        else cout << "no\n";
        break;
    case EQUAL:
        r1.print();
        cout << " == ";
        r2.print();
        cout << " ? ";
        if (r1 == r2)
            cout << "yes\n";
        else cout << "no\n";
        break;
    case NOT_EQUAL:
        r1.print();
        cout << " != ";
        r2.print();
        cout << " ? ";
        if (r1 != r2)
            cout << "yes\n";
        else cout << "no\n";
        break;
    }
    cout << "enter your choice : " << endl;
    cin >> op;
}
return 0;
}
```

שאלה מס' 2:

1. הגדר/י מחלקה MyDate עבור ייצוג תאריך.
המחלקה תכלול את התכונות הבאות:

- יום (int)
- חודש (int)
- שנה (int)

ולכל הפחות, את המתודות הבאות:

- constructor – בנאי המקבל כפרמטרים ערכים עבור התכונות: יום, חודש, שנה, עם ערכי ברירת מחדל - 1/1/1920 (כלומר, במידה ולא התקבל ערך עבור אחד או יותר מהשדות, הפרמטר המתאים יקבל את ערך ברירת המחדל התואם לו). על הבנאי לאתחל את התכונות בערכים שהתקבלו, אך לפני עליו לבצע בדיקת תקינות לתאריך. להלן רשימה של החודשים ומספר הימים שבכל חודש:

מספר חודש	שם חודש	מספר ימים
1	ינואר	31
2	פברואר	28 (29)
3	מרץ	31
4	אפריל	30
5	מאי	31
6	יוני	30
7	יולי	31
8	אוגוסט	31
9	ספטמבר	30
10	אוקטובר	31
11	נובמבר	30
12	דצמבר	31

מספר הימים בחודש פברואר בשנה מעוברת מסומן בסוגריים.
שנה היא מעוברת אם היא מתחלקת ב- 4 אך אינה מתחלקת ב- 100, או שהיא מתחלקת ב- 400.
הקלט לתוכנית צריך להיות בפורמט הבא yyyy mm dd :
כאשר mm, dd ו- yyyy הם מספרים שלמים וחייבים המייצגים יום, חודש ושנה בהתאמה.

הטווחים המותרים הם:

ימים: $1 \leq dd \leq 31$

חודשים: $1 \leq mm \leq 12$

שנים: $0 \leq mm \leq 9999$

שימו לב כי אין זה מספיק שהמספרים נמצאים בטווחים המותרים על מנת שהתאריך יהיה תקין.

לדוגמא: 2000 4 31 - אינו תאריך תקין מכיוון שבאפריל יש 30 ימים. וכן הלאה ...
כאמור, אם התאריך המתקבל כקלט אינו תאריך תקין, על התכנית להדפיס הודעה שהתאריך אינו תקין "Error date", ולאתחל את היום ב 1, החודש ב 1, והשנה ב 1920.

- copy constructor – בנאי העתקה.
- מתודה setDate המעדכנת את התאריך. על המתודה לקבל כפרמטרים ערכים עבור כל התכונות. במידה והערכים תואמים את הטווחים התקינים (כאמור במתודה הבונה), על המתודה להציב את הערכים החדשים בתכונות. במידה ואחד הערכים אינו תואם לטווח המתאים לעיל, לא יתבצע כל שינוי **בכל התכונות**, ללא שום פלט.
- מתודה print אשר מדפיסה את התאריך בפורמט: dd/mm/yyyy (כאשר היום והחודש בני ספרה אחת או שתיים, והשנה בת ארבע ספרות).

- `operator=`. המתודה עובדת כמו האופרטור `=` הטבעי. מעתיקה את ערכי התכונות של האובייקט מצד ימין של הסימן שווה לתכונות של האובייקט בצד שמאל של הסימן שווה.
- `++ operator` **תחילי**. המתודה משנה את התאריך למחרתו (כלומר - לתאריך של יום למחרת), ומעדכנת את התכונות בהתאם. שימו לב שבסוף החודש, יום המחרת הוא היום הראשון של החודש הבא, ובסוף השנה – יום למחרת הוא היום הראשון של השנה הבאה לאחריה.
- `++ operator` **סופי**. המתודה משנה את התאריך למחרתו (כלומר - לתאריך של יום למחרת), ומעדכנת את התכונות בהתאם. (שימו לב גם כאן לתאריכים שבסוף החודש יום המחרת הוא היום הראשון של החודש הבא, ובסוף השנה – יום למחרת הוא היום הראשון של השנה הבאה לאחריה).
- `operator >`. המתודה בודקת קדימות בין שני תאריכים. במידה והתאריך שמשמאל לאופרטור מאוחר יותר מהתאריך שממיין לאופרטור, המתודה תחזיר אמת, אחרת המתודה תחזיר שקר.
- `operator <`. המתודה בודקת קדימות בין שני תאריכים. במידה והתאריך שמשמאל לאופרטור מוקדם יותר מהתאריך שממיין לאופרטור, המתודה תחזיר אמת, אחרת המתודה תחזיר שקר.
- `operator ==`. עבור ערכים בוליאניים יש להדפיס `true` או `false`.

כתוב/י תכנית ראשית אשר תקלוט מהמשתמש תאריך ("**Enter a date**") (שים לב לפורמט **הקלט**). על התכנית לאתחל אובייקט שהוא מופע של המחלקה `MyDate` בתאריך שהתקבל. התכנית תדפיס את התאריך (באמצעות המתודה שהגדרת). לאחר מכן, בלולאה – התכנית תדפיס: "**Enter a code**", תקלוט מהמשתמש קוד פעולה, תבצע את הפעולה המתאימה לקוד בהתאם לרשום להלן, ואז תדפיס את ערכו של האובייקט. שים/י לב! כדי שתוצאות הפעלת האופרטור `++` יודפסו – יש לבצע השמה של תוצאת הפעולה לתוך משתנה חדש. ולאחר מכן להדפיס את ערכו של המשתנה החדש. לדוגמא עבור הפעולה של `++` תחילי יש לבצע:

```
MyDate d1, d2;
d1 = ++d2;
d1.print();
cout << endl;
d2.print();
cout << endl;
```

התכנית תסתיים כאשר ייקלט הקוד 0 (אפס).
קודי הפעולה:

- 1 - עדכון תאריך. התכנית תדפיס **Enter a date**, ותקלוט תאריך חדש מהמשתמש. התכנית תפעיל את המתודה `setDate`. לאחר מכן התכנית תדפיס את התאריך המעודכן.
- 2 - התכנית תדפיס את פעולת האופרטור `++` תחילי. (עיין הערה להלן)
- 3 - התכנית תדפיס את פעולת האופרטור `++` סופי. (עיין הערה להלן)
- 4 - התכנית תדפיס **Enter a date**, ותקלוט מהמשתמש תאריך לתוך עצם חדש. לאחר מכן, התכנית תפעיל את פעולת ההשמה (אופרטור `=`) - ותציב את העצם החדש בתוך העצם הישן (העצם החדש יופיע מצד ימין של אופרטור ההשמה והעצם הישן יופיע מצד שמאל של אופרטור ההשמה לאחר כך יודפס העצם הישן (עם הערכים החדשים שבו, כמובן).
- 5 - התכנית תדפיס **Enter a date**, ותקלוט תאריך מהמשתמש לעצם חדש. התכנית תדפיס את פעולת האופרטור **>** בין התאריך החדש לתאריך הישן (העצם הישן משמאל לאופרטור) **ואז נקודתיים** ואז תוצאת הפעולה (`false` או `true`).
- 6 - התכנית תדפיס **Enter a date**, ותקלוט תאריך חדש מהמשתמש. התכנית תדפיס את פעולת האופרטור **<** בין התאריך החדש לתאריך הישן (התאריך הישן משמאל לאופרטור) **ואז נקודתיים** ואז תוצאת הפעולה (`false` או `true`).

- 7 התכנית תדפיס `Enter a date` , ותקלוט תאריך חדש מהמשתמש.
התכנית תדפיס את פעולת האופרטור `==` בין התאריך החדש לתאריך
הישן (התאריך הישן משמאל לאופרטור) ואז **נקודתיים** ואז תוצאת
הפעולה (`true` או `false`).
- 0 התכנית תפסיק את ביצוע הלולאה, ותסתיים.

דוגמא להרצת התוכנית: (אין להדפיס את ההערות אלא רק את הערכים)

```
Enter a date
-5/1/2012//קלט מהמשתמש
Error date
1/1/2012//הדפסת הערך שהוצב בתוך התאריך
Enter a code
1
Enter a date
5/7/2010//קלט מהמשתמש
5/7/2010//הדפסת התאריך אחרי השינוי
Enter a code
2
6/7/2010//הדפסת אופרטור ++ תחילי על התאריך
6/7/2010
Enter a code
3
6/7/2010//הדפסת אופרטור ++ סופי על התאריך
7/7/2010
Enter a code
4
Enter a date
14/7/2010
14/7/2010//הכנסת תאריך חדש על ידי השמה
Enter a code
5
Enter a date
14/7/2010
14/7/2010 > 14/7/2010 : false
Enter a code
7
Enter a date
14/7/2010
14/7/2010 == 14/7/2010 : true
Enter a code
0
```

בהצלחה רבה!