פעולות על משתנים

תרגיל :

בישראל נהוג למדוד טמפרטורה במעלות צלסיוס, ואילו בארה"ב במעלות פרנהייט. תייר ישראלי המגיע לארה"ב, מתקשה להחליט על-פי תחזית מזג האוויר האם יהיה יום חם או קר.
פתח ויישם אלגוריתם לקליטת טמפרטורה במעלות פרנהייט למשתנה tf. יש להציג כפלט את הטמפרטורה במעלות צלסיוס.

ניתוח הבעיה:

נוסחת המעבר מטמפרטורה במעלות פרנהייט לטמפרטורה במעלות צלסיוס היא:

$$t°c = (t° f - 32) \cdot \frac{5}{9}$$

נפעיל את הנוסחה על דוגמאות קלט שונות:

tf	נוסחת המעבר מפרנהייט לצלסיוס T°F → T°C
95	(95°f - 32) * 5/9 → 35°c
90	(90°f - 32) * 5/9 → 32.222°c
75	(75°f - 32) * 5/9 → 23.88889°c

טבלת משתנים:

מתוך דוגמאות הקלט, רואים שהטמפרטורה המחושבת במעלות צלסיוס יכולה להיות מספר שלם או מספר ממשי (הכולל חלק עשרוני), ולכן נקבע את משתנה הטמפרטורה במעלות צלסיוס להיות double.

שם המשתנה	טיפוס המשתנה	תפקיד המשתנה
tf	int	הטמפרטורה במעלות פרנהייט
tc	double	הטמפרטורה במעלות צלסיוס

אלגוריתם: ***מפרנהייט-לצלסיוס***

{ קלט: הטמפרטורה במעלות פרנהייט.
פלט: הטמפרטורה המחושבת במעלות צלסיוס. }

(1) קלט: טמפ' במעלות פרנהייט ← tf
(2) tc ← (tf – 32) * 5/9
(3) פלט: הטמפ' במעלות צלסיוס: tc

התכנית:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
 *   קלט:  הטמפרטורה במעלות פרנהייט   *
 *   פלט:  הטמפרטורה במעלות צלסיוס   *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
```

```
     import java.util.Scanner;
1.   public class Temperature {
2.           public static void main(String[] args){
3.                   int tf;
4.                   double tc;
5.                   Scanner  input = new Scanner (System.in);
6.                   System.out.println ("Type temperature in Fahrenheit → ");
7.                   tf = input.nextInt();
8.                   tc = (tf - 32) * 5/9;
9.                   System.out.println ("temperature in Celsius is: " + tc);
10.          }
11.  }
```

תוצאות הריצה:

בדיקת תוצאות ריצת התכנית עבור הקלטים שנבדקו ידנית, הראתה את התוצאות הבאות:

קלט ל- tf	**פלט ב- tc**	**תוצאה נכונה**
95	35	✓
90	32	32.222
75	23	23.88889

יתרה מכך - שינוי סדר כתיבת הביטויים בשורה 8 בתכנית: tc = 5/9 * (tf – 32);
יתן תוצאות של 0 מעלות צלסיוס עבור כל ערך קלט.

כדי להבין את מהות השגיאות שהתקבלו, יש להבין את הדרך שבה מבצע המחשב חילוק במספרים שלמים וחילוק במספרים ממשיים. (החישוב מתבצע לפי סדר הופעת האיברים משמאל לימין).

פעולת החילוק

תוצאת פעולת החילוק תלויה בטיפוס הערכים המשתתפים בפעולה. כדי להבטיח תוצאת חילוק מדוייקת (ערך ממשי), יש להקפיד שלפחות אחד הערכים המשתתפים בפעולה יהיה ממשי, וכך גם המשתנה המאחסן את התוצאה:

נבחן את הוראות ההשמה הבאות:

```
int a = 9, b = 4 ;
double x ;
```

```
(1)   x = 9 / 4 ;          //   x ← 2.0      התוצאה שגויה
(2)   x = 9 / 4.0;         //   x ← 2.25
(3)   x = 9.0 / 4;         //   x ← 2.25
(4)   x = (double) a / b ; //   x ← 2.25     {התייחס ל-a כאל double }
```

דוגמא (1) פעולת חילוק של שלם בשלם מחזירה תמיד ערך שלם, ולכן תהיה התוצאה שהתקבלה שגויה. אם נשים את התוצאה שהתקבלה במשתנה מטיפוס ממשי, הוא יישמר כשלם בייצוג ממשי, והתוצאה עדיין תהיה שגויה.

דוגמאות (2) ו- (3) מספיק שאחד הערכים בפעולה יהיה ערך ממשי, כדי שהתוצאה תהיה מטיפוס ממשי. אם המשתנה המקבל אף הוא מטיפוס ממשי, תתקבל תוצאה ממשית.

דוגמא (4) **casting** – **המרה**. רישום (double) לפני הביטוי החשבוני מבהיר למהדר שהתוצאה תהיה מטיפוס ממשי. (יש לרשום את טיפוס הנתונים בתוך סוגריים עגולים).
<u>שים ♥</u>: ההמרה משלם לממשי נעשית על המשתנה a ולא על תוצאת החילוק.
סדר החישוב הוא: ((**double**) a) / b ;).

חשוב: מה תהיה התוצאה של הביטוי: 3/5
אם התוצאה תאוחסן במשתנה מטיפוס שלם: _____ במשתנה מטיפוס ממשי: _____

```
int i;
double d;
```

```
(1)   i = 3/5 ;    //   i ← 0
(2)   d = 3/5 ;    //   d ← 0.0
(3)   d = 3/5.0 ;  //   d ← 0.6
```

הסבר: <u>שורות (1) ו-(2)</u>: התוצאה השלמה של 3/5 היא 0 ולכן i קיבל את הערך השלם 0 ואילו d קיבל את הערך הממשי 0.0.

<u>שורה (3)</u>: התוצאה של 3 / 5.0 היא 0.6 ולכן הושמה תוצאה זו ב- d.

תיקון התכנית להמרת הטמפרטורה מפרנהייט לצלסיוס:

כדי שהתכנית תיתן תוצאה נכונה, יהיה עלינו לגרום לאחד הערכים, בהוראת החישוב שבשורה 12, להיות מטיפוס ממשי.

כל אחת מהאפשרויות הבאות תיתן את התוצאה הנכונה המבוקשת:

הפיכת קבוע מספרי למספר ממשי:

(i) tc = (tf – 32) * 5.0 / 9;
(ii) tc = (tf – 32) * 5 / 9.0;
(iii) tc = (tf – 32.0) * 5 / 9 ;

casting - המרה, התייחסות למשתנה מטיפוס שלם כאל ממשי:

(iv) tc = ((**double**) tf – 32) * 5/9;

מנה ושארית – פעולת חילוק בשלמים

תרגיל: *אבא חיים קנה שקית גולות על מנת לחלקן בין ארבעת ילדיו. כשהגיע הביתה ספר ומצא כי בשקית 13 גולות. כיון שבמשפחה יש שוויון בין הילדים מעוניין אבא חיים לדעת מהו המספר המרבי של גולות שיקבל כל ילד, וכמה גולות יישארו אצלו לאחר החלוקה .*

ניתוח הבעיה:

13 גולות אינן מתחלקות ב- 4. ברור שאבא חיים אינו יכול לתת לכל אחד מבניו 3¼ גולות, ולכן כל ילד יקבל 3 גולות ותישאר אצל אבא חיים גולה אחת אותה לא ניתן לחלק.

על מנת לחשב כמה גולות יקבל כל ילד נחשב את ה**מנה** של חלוקת 13 ב- 4.
על מנת לחשב כמה גולות יישארו אצל אבא חיים נחשב את ה**שארית** של חלוקת 13 ב- 4.

חישוב ה**שארית** נעשה על ידי האופרטור %

n1 ו- n2 הינם משתנים מטיפוס שלם:

- תוצאת הביטוי: n1 / n2 היא **<u>המנה</u>** של החלוקה (ערך שלם).
- תוצאת הביטוי: n1 **%** n2 היא **<u>השארית</u>** של החלוקה (ערך שלם).

טבלת משתנים:

שם המשתנה	טיפוס המשתנה	תפקיד המשתנה
marbles	int	מספר הגולות. המספר **המחולק**.
children	int	מספר הילדים. המספר **המחלק**.
div	int	מספר הגולות שיקבל כל ילד. **המנה**.
mod	int	מספר הגולות ששארו בידו של אבא חיים. **השארית**.

פתרון בעייתו של אבא חיים:

```
int  marbles = 13, children = 4 ;
int  div, mod ;

div   = marbles / children ;
mod = marbles % children ;
```

מעקב אחר פעולות החישוב, מראה שכל ילד יקבל 3 גולות. div ← 3
ובידיו של אבא חיים תישאר גולה אחת שלא ניתנת לחלוקה. mod ← 1

<u>שים ♥</u>: כל המשתנים המשתתפים בפעולות החישוב חייבים להיות מטיפוס int .
פעולת חישוב השארית % <u>אינה</u> משמעותית עבור ערכים מטיפוס ממשי.

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
 *       תכנית הבודקת את הערכים המוחזרים על ידי חילוק ושארית בשלמים          *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
import java.util.Scanner;

public class DivMod {
        public static void main (String[] args){
                int n1, n2;             //  (n2) והמחלק  (n1) המחולק
                int div, mod;           //   (mod) והשארית  (div) המנה

                Scanner = new Scanner (System.in);

                System.out.print ("Type 1st number → ");
                n1 = input.nextInt();

                System.out.print ("Type 2nd number → ");
                n2 = input.nextInt();

                div = n1 / n2 ;
                mod = n1 % n2 ;

                System.out.println (n1 + "/" + n2 + " = " + div);
                System.out.println (n1 + "%" + n2 + " = " + mod);
        }
}
```

נבדוק את הפלט המתקבל בהרצת התכנית הטבלת מעקב עבור 4 דוגמאות קלט מייצגות:

n1	n2	div	mod	מסך / פלט	ריצה
23	4	5	3	Type 1st number → 23 Type 2nd number → 4 23 / 4 = 5 23 % 4 = 3	א. מחלק ומחולק חיוביים.
-23	4	-5	-3	Type 1st number → -23 Type 2nd number → 4 -23 / 4 = -5 -23 % 4 = -3	ב. מחולק שלילי, מחלק חיובי.
23	-4	-5	3	Type 1st number → 23 Type 2nd number → -4 23 / -4 = -5 23 % -4 = 3	ג. מחולק חיובי, מחלק שלילי.
-23	-4	5	-3	Type 1st number → -23 Type 2nd number → -4 -23 / -4 = 5 -23 % -4 = -3	ד. מחלק ומחולק שליליים.

שימושים של פעולות החילוק בשלמים

מחלקים של מספר.

מחלקים של מספר הינם כל המספרים שאם נחלק בהם את המספר, יתנו שארית 0.

לדוגמא: המספר 5 מחלק את כל המספרים המסתיימים ב- 5 או ב- 0 : ... , 25 , 20 , 15 , 10 , 5
המספר 2 מחלק את כל המספרים הזוגיים.

כדי לדעת אם המספר שבמשתנה b מחלק את המספר שבמשתנה a, נבדוק מהי השארית המתקבלת מחלוקת b ב- a : a % b

כדי לדעת האם המספר שב- num הוא מספר זוגי, נבדוק מהי שארית החלוקה של num ב- 2 :
שארית 0 - num זוגי.
שארית 1 - num אי-זוגי.

בפרקים הבאים נעסוק בהרחבה בבדיקת התחלקות ומציאת מחלקים של מספר.

חלוקה לקבוצות.

תרגיל: *מפעל לייצור כפתורים אורז כל 12 כפתורים באריזת ניילון וכל 50 אריזות בקופסה. פתח אלגוריתם לקליטת מספר הכפתורים שייוצרו ביום מסויים, ויציג כפלט את מספר הקופסאות המלאות, מספר האריזות הבודדות שנשארו ולא מילאו קופסה, ואת מספר הכפתורים הבודדים שלא מילאו אריזה.*

טבלת משתנים:

שם המשתנה	טיפוס המשתנה	תפקיד המשתנה
buttons	int	מספר הכפתורים שייצר המפעל ביום מסויים.
pack	int	מספר האריזות המלאות.
fullBox	int	מספר הקופסאות המלאות.
singles	int	מספר הכפתורים הבודדים שלא מילאו אריזה.
remainPack	int	מספר האריזות שלא מילאו קופסה.

שים ♥: כל המשתנים מטיפוס שלם.

ניתוח הבעיה ודרך החישוב:

חלוקת buttons ב- 12 ייתן את מספר האריזות המלאות. pack ← buttons / 12
חישוב השארית של buttons ב-12 ייתן את מספר הכפתורים שלא מילאו אריזה מלאה. singles ← buttons % 12

חלוקת מספר האריזות ב- 50 ייתן את מספר הקופסאות המלאות. fullBox ← pack / 50
חישוב השארית של מספר האריזות ב- 50 ייתן את מספר האריזות שלא מילאו קופסה. remainPack ← pack % 50

פרוק מספר לספרותיו.

תרגיל: *פתח ויישם אלגוריתם לקליטת מספר דו ספרתי חיובי והצגה כפלט מספר דו ספרתי חדש המורכב מאותן ספרות בסדר הפוך. לדוגמא: אם יתקבל כקלט המספר 25, יוצג כפלט המספר 52.*

ניתוח הבעיה:

מספר דו-ספרתי מורכב משתי ספרות – ספרת אחדות וספרת עשרות - במספר 25 ספרת האחדות היא 5 וספרת העשרות היא 2. במספר 52 תהיה ספרת האחדות 2 וספרת העשרות 5.

נשים לב שלא כל מספר דו-ספרתי יכול להיות דוגמא מייצגת – מספר שספרת האחדות שלו היא 0 אינו יכול להיחשב כדוגמא מייצגת, ובאותה מידה לא ננסה להפוך ספרותיו של מספר שלילי למרות שהאלגוריתם נכון גם עבורו.

פירוק הבעיה לתת-בעיות:

תת-בעיה 1: קלט מספר דו-ספרתי.
(בשלב הראשון נסתפק בפעולת קלט מן המשתמש. בשלב מתקדם יותר נוסיף לשלב זה בדיקות תקינות לקלט שהתקבל).

תת-בעיה 2: עיבוד: מתוך התבנית המתמטית מתברר כי כדי להפוך את ספרות המספר, יש לפרק תחילה את המספר לספרותיו ואחר כך להרכיב את המספר החדש מספרות המספר המקורי.
פירוק המספר יעשה באמצעות הפעולות חלוקה ב- 10 ושארית החלוקה ב- 10.
הרכבת המספר החדש תיעשה על ידי שימוש בתבנית המספר הדו-ספרתי.
תבנית מספר דו ספרתי המורכב מספרת האחדות b ומספרת העשרות a: $10a + b$.

תת-בעיה 3: פלט: הצגת המספר החדש.

בחירת משתנים:

שם משתנה	טיפוס נתונים	תפקיד המשתנה
num	int	המשתנה לתוכו נקלט המספר הדו-ספרתי. (num קיצור של number).
newNum	int	המשתנה לתוכו יוכנס המספר החדש.
d1	int	המשתנה שיכיל את ספרת האחדות. (d עבור digit – ספרה).
d10	int	המשתנה שיכיל את ספרת העשרות.

שים ♥: בחירת שמות משמעותיים למשתנים תורמת לבהירות האלגוריתם.

האלגוריתם: ***מספר-דו-ספרתי***

{ קלט: num מספר דו-ספרתי.
פלט: newNum מספר דו-ספרתי המורכב מספרות המספר num בסדר הפוך. }

(1) קלט: מספר דו-ספרתי ← num
(2) d1 ← num % 10 { ספרת האחדות }
(3) d10 ← num / 10 { ספרת העשרות }
(4) newNum ← d1 * 10 + d10
(5) פלט: newNum

בדיקת נכונות האלגוריתם באמצעות טבלת מעקב:

נבדוק את נכונות האלגוריתם עבור הדוגמא.

num	d1	d10	newNum	פלט
25	5	2	52	52

נוכל להריץ את האלגוריתם בטבלת מעקב עבור דוגמאות קלט נוספות כדי לוודא את נכונותו.

יישום האלגוריתם:

ניישם את האלגוריתם לתכנית מחשב, ונקפיד להוסיף תיעוד והערות הסבר לחישובים שביצענו כדי שבעתיד, אם נרצה לשנות ולהוסיף לתכנית, נוכל לזהות את החלקים השונים.

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
 *                                  קלט:  מספר דו-ספרתי.          *
 *          פלט:  מספר דו-ספרת חדש המורכב מאותן ספרות בסדר הפוך.          *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */

import java.util.Scanner;

public class Numbers {
    public static void main(String[] args){
            int  num, newNum ;              //  המספר והמספר החדש
            int  d1, d10 ;                  //  ספרות המספר

            Scanner input = new Scanner (System.in);

            System.out.print ("type a 2 digit number → ") ;
            num = input.nextInt () ;

            //---  פרוק המספר לספרותיו   ---
            d1  = num % 10 ;                //  ספרת האחדות
            d10 = num / 10 ;                //  ספרת העשרות

            newNum = d1*10 + d10 ;

            System.out.println ("The reversed number is: " + newNum);

    }

}
```

מה יקרה אם נרצה לפרק מספר בן 3 ספרות או יותר?

פרוק מספר תלת ספרתי:

נפתור את התרגיל הקודם עבור מספר תלת-ספרתי:

```
(1) קלט: מספר תלת-ספרתי → num
(2) d1 ← num % 10          { ספרת אחדות }
(3) temp ← num / 10        { ב- temp יש עתה מספר דו-ספרתי }
(4) d10 ← temp % 10        { ספרת עשרות }
(5) d100 ← temp / 10       { ספרת המאות }
(6) newNum ← d1 * 100 + d10 * 10 + d100   { הרכבת המספר בסדר ספרות הפוך }
(7) פלט: newNum
```

נריץ את האלגוריתם בטבלת מעקב:

num	d1	temp	d10	d100	newNum	פלט
374	4	37	7	3	473	473

<u>הערות</u>:

- temp (קיצור של temporary) הוא משתנה עזר זמני. במשתנה זה נשמור תוצאות ביניים של החישובים. בסיום החישוב אין בו יותר צורך, וגם לא נציג את ערכו בפלט.
- אחרי הסרת ספרת האחדות מ- num, נשאר ב- temp מספר דו-ספרתי.
  נפרק את הערך שב- temp כפי שלמדנו לפרק מספר דו-ספרתי.
- ניתן היה להתחיל את הפרוק מצד שמאל של המספר, מהספרה המשמעותית ביותר שלו, ולקבל תוצאה דומה. כך או כך, מומלץ לעבוד בצורה שיטתית כדי שלא להסתבך בפרוק מספרים בעלי מספר רב יותר של ספרות.

<u>שים ♥</u>:

- חובה להצהיר על המשתנה temp אם עושים בו שימוש.
- חובה להציג בטבלת המעקב עמודה עבור המשתנה temp.

פירוק מספר 4 ספרתי:

```
(1) d1 ← num % 10          { ספרת אחדות }
(2) temp ← num / 10        { ב- temp יש עכשיו מספר תלת-ספרתי. }
(3) d10 ← temp % 10        { ספרת עשרות }
(4) temp ← temp / 10       { ב- temp יש עכשיו מספר דו-ספרתי. }
(5) d100 ← temp % 10       { ספרת המאות }
(6) d1000 ← temp / 10      { ספרת האלפים }
(7) newNum ← d1 * 1000 + d10 * 100 + d100 * 10 + d1000
(8) פלט: newNum            { הרכבת המספר בסדר ספרות הפוך }
```

- תחילה נבודד את ספרת האחדות. לאחר שחילקנו את המספר ב- 10, נשאר ב-temp מספר תלת-ספרתי, שאותו כבר למדנו לפרק.

סדר החישוב בביטויים חישוביים

- באופן כללי, החישוב מתבצע משמאל לימין.
- אם מופיעים בביטוי סוגריים, יחושב תחילה הביטוי בתוך הסוגריים.
- עפ"י סדר הקדימויות המתמטי: פעולות החישוב * , / , % מתבצעות לפני הפעולות + , -

טיפוס התוצאה

כאשר יש ביטוי חשבוני מעורב, חשוב לדעת מראש מאיזה טיפוס תהיה התוצאה:

- ערכו של ביטוי חשבוני המכיל לפחות ערך ממשי אחד - הוא ממשי.
- ערכו של ביטוי חשבוני המכיל רק ערכים שלמים - הוא שלם.
- הצבת הסימנים +/- לפני ביטוי חשבוני אינה משנה את טיפוס הביטוי.

הביטוי:	החישוב:	התוצאה	ערך התוצאה:
4 * (8 / 2.0)	4 * 4.0	16.0	ממשי
9 / 2 + 3	4 + 3	7	שלם
(double) 9 / 2 +3	4.5 + 3	7.5	ממשי
5 + 6.8 - 2	11.8 - 2	9.8	ממשי
(5.5 + 0.5) / 2	6.0 / 2	3.0	ממשי
8 % 3 + 1	2 + 1	3	שלם
9 % (2 + 1)	9 % 3	0	שלם
(4 * 3 + 9) / 2 – 6.0	(12 + 9) / 2 – 6.0 21/2-6.0 ⇦ 10-6.0 ⇦	4.0	ממשי
(4 * 3 + 9) / 2.0 – 6	(12 + 9) / 2.0 – 6 21/2.0 - 6 ⇦ 10.5 - 6 ⇦	4.5	ממשי

המרות בין שלמים ומשויים

בפרק 2, הצגנו רק חלק מהמשתנים המספריים הקיימים בשפת Java. לכל טיפוס נתונים יש מספר שונה של בתים בזיכרון וצורת ייצוג בהתאם. למשל, טווח הערכים של מספר (מכוון) מטיפוס **int** הוא 2 ± מיליארד לערך, ואילו טווח הערכים של מספר (לא מכוון) מטיפוס **uint** (unsigned int) הוא בין 0 ל- 4 מיליארד לערך, וזאת למרות ששניהם תופסים בדיוק 4 בתים בזיכרון.

בתחילת פרק זה, ראינו שניסיון לבצע השמה של ערך מטיפוס אחד לתוך משתנה מטיפוס אחר עלול להיות כרוך באובדן מידע.

נבחן את נושא ההמרות בין משתנים מטיפוס שלם וממשי.

המרה ללא הפסד מידע

קבוצת הממשיים כוללת את כל המספרים השלמים.
כדי להמיר ממספר שלם למספר ממשי, מספיק שנוסיף נקודה העשרונית לשלם ויתקבל מספר ממשי.
בהמרה מסוג זה אין אבדן מידע, והיא נקראת **המרה מרומזת (Implicit Conversion)**.

דוגמא 1:

חישוב הממוצע של שני מספרים שלמים על ידי חלוקה בקבוע מספרי שלם שהומר לממשי.

```
int grd1 = 80, grd2 = 75;
double average = (grd1 + grd2) / 2.0 ;
```

ב- average יהיה הערך 77.5

דוגמא 2:

השמת הערך של משתנה שלם במשתנה הממשי:

```
int i = 74;
double d = i;
System.out.println ("d = " + d);
```

הפלט שיתקבל יהיה 74.0

המרה עם אפשרות להפסד מידע

ניסיון להמיר ערך ממשי לשלם, כרוך באבדן המידע של החלק השבור. למשל, המרת 54.92 לשלם תיתן את התוצאה 54. עבור תלמיד שזהו ציונו בבחינה, כשהציון העובר הוא 55, הפסד המידע הוא משמעותי ביותר.

בהמרה מממשי לשלם מתבצע קיטום של החלק השבור.

נדגיש כי ללא ביצוע המרה, לא ניתן לבצע השמה של ערך או משתנה ממשי למשתנה שלם. על ניסיון השמה מסוג זה נקבל הודעת שגיאה כבר בשלב הקומפילציה:

Cannot implicitly convert type 'double' to 'int'

כדי לבצע את ההמרה, יש לרשום לפני הערך להמרה את טיפוס הנתונים החדש:

דוגמא:

```
double sum = 791.35;
int total = (int) sum ;          // total ← 791
```

דוגמא : תכנית הקולטת מספר ממשי, ומפרקת אותו לחלק השלם (trunc) ולחלק השבור (frac).

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
 *   תכנית המפרקת מספר ממשי לחלק השלם ולחלק השבור שלו   *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

import java.util.Scanner;

public class Fractions {
        public static void main(String[] args){
                int trunc;
                double  frac, x ;

                Scanner input = new Scanner (System.in);

                System.out.print ("Type a float number → ");
                x = input.nextDouble();

                trunc = (int) x ;                    //  החלק השלם
                frac  = x - trunc ;                  //  החלק השבור

                System.out.println ("trunc of  " + x + "  =  " + trunc) ;
                System.out.println ("frac of  " + x + "  =  " + frac) ;
        }
}


/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 Type a float number    → 234.123456              :  ריצה 1
 trunc of 234.123456  =  234
 frac of 234.123456  =  0.123456


 Type a float number    → 1.2567                              :  ריצה 2
 trunc of 1.2567  =  1
 frac of 1.2567  =  0.2567

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
```

תרגום ביטויים מתמטיים לשפטים בשפה

תכניות מחשב נועדו לביצוע חישובים מורכבים. לעיתים החישוב שיש לבצע מוצג בכתיבה מתמטית כשבר שיש לו מונה ומכנה מורכבים. נשים לב שללא המרה, מרומזת או מפורשת, אנו עלולים לקבל תוצאות לא מדויקות.

דוגמא 1: יש לחשב בתכנית את ערך הביטוי הבא למשתנה x: $x \leftarrow \frac{3a-5b}{a+b}$
a ו- b הם משתנים בתכנית ויש בהם ערכים מספריים.

בשפות תכנות גם ביטוי חשבוני מסוג שבר ירשם בשורה אחת. כדי להבטיח שסדר הקדימויות המתמטי יישמר, נתייחס לכל הביטוי כאילו נרשם: **x = (double) (מונה) / (מכנה) ;**

נזכור לרשום את פעולת הכפל עליה "ויתרנו" בכתיבה אלגברית.

הביטוי בשפת תכנות: $x=(3.0*a-5*b)/(a+b);$ או $x=(double)(3*a-5*b)/(a+b);$

דוגמא 2: כיצד נתרגם ביטוי מסוג: $x \leftarrow \frac{-b}{2a}$ שבו המכנה הוא מכפלה.

אם נרשום: $x=-b/2*a;$, הרי שבגלל סדר הקדימויות, חישבנו את הביטוי השגוי: $x \leftarrow \frac{-b}{2}*a$

הכתיבה הנכונה צריכה להיות: $x=-b/(2*a);$

<u>שים ♥</u>: אם המשתנים a ו- b הם מטיפוס שלם, תתקבל ב- x תוצאה לא מדויקת, לכן נמיר את התוצאה לממשי, על ידי התייחסות לאחד המשתנים כערך ממשי:

ההמרה תתבצע על ידי casting לאחד המשתנים: $x=((double)-b)/(2*a);$
או הפיכת קבוע מספרי כלשהו המופיע בתרגיל לקבוע ממשי: $x=-b/(2.0*a);$

דוגמא 3: יש לכתוב הוראה בשפה שתחשב את ערך הביטוי הבא: $y \leftarrow \frac{a+3d}{5c}+\frac{3c}{a+c}+3a$

בהנחה שלפחות אחד המשתנים a או c הם משתנים מטיפוס ממשי:

$y=(a+3*d)/(5*c)+(3*c)/(a+c)+3*a;$

אחרת נבקש להתייחס לאחד המשתנים כממשי, או – אם יש קבוע מספרי, נרשום אותו כממשי.

דוגמא 4: לסוגרים מרובעים וסוגרים מסולסלים, המקובלים בכתיבה מתמטית, יש תפקיד מיוחד בשפה. ולכן, בתרגום של ביטוי מתמטי המכיל כמה סוגי סוגריים, נשתמש אך ורק בסוגריים עגולים, תוך הקפדה על שמירה של סוגריים מאוזנים.

הביטוי: $m \leftarrow \left[3(a+b)-\frac{b}{2a}\right]+5$ ירשם כ-: $m=(3*(a+b)-b/(2*a))+5;$

תרגילים

1. לגבי כל אחת מההוראות הבאות, ציין אם היא תקינה. אם לא, ציין מדוע.
הצע תיקון להוראות השגויות.

```
int  a, b, c  ;
double  x, y, z ;
```

1. x = x + 1 ; _____
2. a = y + 2 ; _____
3. a = a - 2 ; _____
4. a / b = z ; _____
5. y = c - 5 ; _____
6. b = c / 2 ; _____
7. c = a * b ; _____
8. y ← y + 5 ; _____
9. b = z * c ; _____
10. c = x % y ; _____
11. a = z ; _____
12. z = 1/a ; _____
13. a + b = c + b ; _____
14. d = c + 0.5 ; _____
15. x = a / b ; _____
16. z = a / b ; _____
17. a = c % 5 ; _____
18. b = x / 7 ; _____
19. c *= c * b ; _____
20. x = a % y ; _____

2. כתוב ביטוי מתמטי לכל אחת מפקודות Java הבאות: (הנח שהמשתנים הנם מטיפוס double)

ביטוי בשפת Java	ביטוי מתמטי
(1) x = b / c + d;	_____
(2) x = b / (c + d);	_____
(3) y = a+(b*c-d) / z+1	_____
(4) z = a*b*c / (d+3)	_____
(5) w = (d-a / b) / ((r-1)-1)	_____
(6) y = (a*(b*c-d)) / (z+1)	_____
(7) v = (d-a / (b+c)) / (r-1)	_____

3. כתוב ביטוי בשפת Java לכל אחד מהביטויים המתמטיים הבאים (קבע משתנה לתוכו תוכנס התוצאה):

ביטוי מתמטי	ביטוי בשפת Java
(1) $x = \frac{a+b}{c*2+z} * (c-d)$	_____
(2) $y = \frac{z}{c-d} * t - k$	_____
(3) $k = \frac{(3x-7)(4y+5)}{2xy}$	_____
(4) $u = \frac{x^2-(x-3)y+1}{4x-3y}$	_____

מנה ושארית – חילוק בשלמים

4. בין אלו סוגי משתנים ניתן לבצע פעולת חילוק / ? _____

בין אלו סוגי משתנים ניתן לבצע פעולת חישוב שארית % ? _____

5. מהו סדר הקדימויות בפעולות / ו- %? _____

חשב: 10 % 10 / 794 _____

חשב: 10 % (10 / 794) ? _____

האם התוצאות שקולות או שונות ? _____

6. האם כשמבצעים פעולות / ו- % בלבד ניתן לקבל תוצאה לא שלמה? _____

נמק! _____

7. השלם את הטבלה הבאה. (הנח שהמשתנים הינם מטיפוס int):

x	y	x / y	x % y
13	3		
13	-3		
-13	3		
-13	-3		
4	9		
0	5		
18	6		
18	4		
74	10		

8. מה תוצאות הביטויים הבאים כאשר ערכו של x הינו 1874.

(1) x % 10 _____ (7) x / 10 % 100 _____

(2) x / 10 _____ (8) x / 1000 _____

(3) x % 100 _____ (9) x % 1000 _____

(4) x / 100 _____ (10) x / 10 % 10 _____

(5) x % 100 / 10 _____ (11) x % 10 / 10 _____

(6) x % (100 / 10) _____ (12) x % (10 / 10) _____

חשוב: ידוע ש: x % 5 = 3 וכן: 15 ≤ x ≤ 27. אלו ערכים יכולים להיות ב- x ?

9. לפניך קטע תכנית. המשתנים num , d1 ו- d10 מטיפוס שלם (int).

```
System.out.println ("Enter a number between 10 to 99 “);
num = input.nextInt();
d1  = num % 10;
d10 = num / 10;
System.out.println ("num = " + d10*10 + d1);
System.out.println ("new num = " + d1*10 + d10);
```

פלט / מסך	d10	d1	num

עקוב אחר קטע התכנית בטבלת מעקב, ורשום:

א. מהו הפלט המתקבל עבור הקלט 73 ?

ב. מהו הפלט המתקבל עבור הקלט 90 ?

פיתוח ויישום אלגוריתם

10. פתח ויישם אלגוריתם הקולט שלושה מספרים שלמים לתוך משתנים a, b ו- c, ומחשב ומדפיס את הממוצע (המדוייק) שלהם.

11. פתח ויישם אלגוריתם לקליטת אורך הצלע של ריבוע. יש להציג כפלט את שטח הריבוע ואת היקפו.

12. פתח ויישם אלגוריתם לקליטת אורך הצלעות של מלבן. יש להציג כפלט את שטח המלבן ואת היקפו.

13. פתח ויישם אלגוריתם לקליטת אורך הצלע של ריבוע. יש לחשב את שטח הריבוע. לאחר מכן יש להגדיל את אורך צלע הריבוע ב- 3 ולחשב את שטח הריבוע החדש. יש להציג כפלט את שטחי שני הריבועים ואת הפרש השטחים.

14. פתח ויישם אלגוריתם לקליטת אורך הצלעות של מלבן וחישוב שטחו. לאחר מכן יש להגדיל את אורך המלבן פי 2 ולהקטין את רוחבו ב-1 ולחשב את שטח הריבוע החדש. יש להציג כפלט את שטחי שני המלבנים ואת הפרש השטחים.

15. פתח ויישם אלגוריתם לקליטת אורך הבסיס של משולש, אורך שתי הצלעות האחרות, והגובה לבסיס. יש להציג כפלט את שטח המשולש ואת היקפו.

16. פתח ויישם אלגוריתם לקליטת אורך הבסיסים של טרפז, אורך שתי הצלעות האחרות, והגובה. יש להציג כפלט את שטח הטרפז ואת היקפו.

מנה ושארית

17. פתח ויישם אלגוריתם הקולט שלושה מספרים שלמים לתוך משתנים a, b ו- c, ומחשב ומדפיס את שארית החלוקה של a בסכום של b + c.

18. פתח ויישם אלגוריתם לקליטת מספר דו-סיפרתי והדפסת ספרת העשרות וספרת האחדות שלו.
הנחייה: יש לקלוט מספר דו-סיפרתי ולפרקו באמצעות הפעולות המחזירות מנה ושארית.

19. פתח ויישם אלגוריתם לקליטת מספר תלת-סיפרתי והדפסת המספר החדש שנוצר מאותן ספרות בסדר הפוך. (שים ♥! 3 ספרות בודדות אינן מהוות מספר-תלת ספרתי).
לדוגמא: אם נקלט המספר 736 יודפס המספר 637.

20. פתח ויישם אלגוריתם לקליטת מספר דו-ספרתי ומספר חד-ספרתי. יש ליצור ולהדפיס מספר תלת-ספרתי הנוצר מהכנסת המספר החד ספרתי בין ספרות המספר הדו-ספרתי.
לדוגמא: אם קלטים המספרים 78 ו- 3, יודפס המספר 738.

21. פתח ויישם אלגוריתם לקליטת מספר ארבע-ספרתי והדפסת המספר הנוצר מהיפוך ספרות המספר. לדוגמא: אם נקלט המספר 1234 יודפס המספר 4321.

22. פתח ויישם אלגוריתם לקליטת מספר ארבע ספרתי ויצירת "סיבוב מעגלי" אחד על המספר, על ידי הפיכת הספרה הכי פחות משמעותית במספר (LSD – Less Significant Digit) לספרה המשמעותית ביותר (MSD – Most Significant Digit). לדוגמא: אם נקלט המספר 1234, יודפס המספר 4123 (ספרת האחרות הפכה להיות ספרת האלפים).

23. פתח ויישם אלגוריתם לקליטת מספר ארבע-ספרתי והדפסת המספר הנוצר מהחלפת שתי הספרות הימניות בשתי הספרות השמאליות שלו. לדוגמא: אם נקלט המספר 1234 יודפס המספר 3412. (בכמה דרכים שונות ניתן לבצע את המשימה ?)

24. כתה בת n תלמידים יצאה לטיול. התלמידים התבקשו להתחלק לקבוצות בנות 5 תלמידים כל אחת. פתח ויישם אלגוריתם לקליטת מספר התלמידים בכיתה והדפסת מספר הקבוצות שנוצרו. כמו כן יש להדפיס את מספר התלמידים שנשארו ללא קבוצה.

אלגוריתמים לחישובים מיוחדים

25. פתח ויישם אלגוריתם לקליטת מספר ממשי והדפסת החלק העשרוני שלו .
לדוגמא: אם נקלט המספר 4.65 יודפס הערך 0.65

26. פתח ויישם אלגוריתם לקליטת מספר ממשי. יש לחשב ולהדפיס את ערכו המעוגל לשלם הקרוב.
דוגמא: המספר 3.25 יעוגל ל- 3, המספר 7.5 יעוגל ל- 8, המספר 1.914 יעוגל ל- 2.

27. פתח ויישם אלגוריתם לקליטת מספר ממשי. יש לחשב ולהדפיס:
   א. המספר המעוגל עד לדיוק של ספרה אחת אחרי הנקודה העשרונית.
   ב. המספר המעוגל עד לדיוק של שתי ספרות אחרי הנקודה העשרונית.
   ג. המספר המעוגל עד לדיוק של שלוש ספרות אחרי הנקודה העשרונית.

<u>שים ♥</u>: אין להסתפק בהצגת המספר בפורמט הדפסה מתאים אלא יש לקצץ את החלק המיותר במספר.

28. נתונה התכנית הבאה:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
 *                                                                     *
 *      קלט: מספר ממשי                                                *
 *      פלט: _____                                                    *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/

import java.util.Scanner;

public class RoundNumber {
        public static void main(String[] args){
                int roundNum;
                double  x;

                Scanner input = new Scanner (System.in);

                System.out.print ("Type a float number → ");
                x = input.nextDouble();

                roundNum = (int) (x + 0.5);                                    (*)

                System.out.println ("x = " + x + ",   roundNum = " + roundNum);

        }
}
```

א. הרץ את התכנית בטבלת מעקב והשלם שורת תאור הפלט.

ב. שנה את התכנית כך שהיא תהיה מסוגלת לטפל גם במספרים שליליים.

ג. **חשוב**: מה תהיה התוצאה אם נחליף את השורה המסומנת ב- (*) בקטע הבא:

```
x = x * 10;
roundNum = (int) (x + 0.5);
x = roundNum / 10.0;
```

<u>שים ♥:</u> ההמרה לשלם נעשית על כל הביטוי

ד. מה תהיה התוצאה ב- x אם נחזור על תהליך זה אך נכפול ונחלק במאה? באלף?

שגיאות מתכנית

כבר בשלבים הראשונים של העבודה עם המחשב, יש סיכוי שניתקל בהודעות שגיאה, ולכן כדאי שנכיר את השגיאות הנפוצות ונלמד לזהות אותן.

שגיאה אין פירושה טעות או דבר שטות, אלא פשוט דבר שהמחשב אינו מבין. המחשב אינו יכול לנחש את כוונת המתכנת.
קיימים כללי תחביר אשר עליהם יש להקפיד. אי הקפדה על כללים אילו גוררת אחריה הודעת שגיאה גם אם כוונת המתכנת הייתה נכונה.
הודעות השגיאה מצביעות על מגבלות המחשב יותר מאשר על מגבלות המתכנת.

בתכנות קיימים שלושה סוגי שגיאות:
1. שגיאות הקשורות ב**כללי התחביר** של השפה.
2. שגיאות **זמן ביצוע**, המתגלות בזמן ריצת התכנית (בעקבות ביצוע פעולה מסויימת).
3. שגיאות **לוגיות** שהן בעצם שגיאות כוונה.

שגיאה תחבירית

שגיאה זו נגרמת כאשר ביצענו שגיאת כתיב בהקלדת הוראה, בהקלדת שם של משתנה, או כאשר משתמשים בתחביר לקוי או בסימני פיסוק לא מדויקים.

חלק מהשגיאות מתגלות כבר בשלב ההקלדה על ידי העורך המסמן את מקום השגיאה, וחלקן תתגלינה רק בשלב ההידור. ברשימת השגיאות יירשם לכל שגיאה באיזו שורה היא נמצאה, ומה מהות השגיאה.

דוגמאות שכיחות:

- **שגיאת כתיב:** כתיבת של הוראה בשפה בצורה שגויה.
  בזמן הידור התכנית (קומפילציה) תופיע ההודעה:

cannot resolve symbol

- **משתנה לא מוגדר:**
  שגיאה מסוג זה יכולה להתרחש כאשר לא הגדרנו משתנה או כאשר שגינו בהקלדת שם המשתנה.
  בזמן ההידור תופיע ההודעה:

cannot resolve symbol

- **אין ; (נקודה פסיק) בסוף ההוראה:**
  כבר בשלב הכתיבה, יופיע סימן שגיאה בסוף השורה.
  בשלב ההידור תופיע ההודעה: *';' expected*
  בצירוף מספר השורה בה זוהתה השגיאה.

- **השמטה של תו הגרשיים לפני או בסיום מחרוזת:**
  בזמן ההידור תופיע ההודעה: *unclosed string literal*

- **אין { (סוגריים מסולסלים) בסוף התכנית:**
  בזמן ההידור תופיע ההודעה: *'}' expected*

שגיאת זמן ביצוע

שגיאות הנגרמות כאשר הקומפיילר אינו מסוגל לבצע את ההוראה בהצלחה בשל תנאים לא מתאימים. השגיאה הנפוצה היא שגיאה של ניסיון לחלק באפס.

שגיאות לוגיות – שגיאות כוונה

שגיאה הנגרמת כאשר הסתיים ביצוע התכנית, אולם התבצע משהו השונה מהמשימה אליה התכוונו. השגיאה הנפוצה היא לולאה אינסופית הנגרמת בגלל תנאי סיום לולאה שלא התממש.
הקומפיילר אינו יכול לזהות שגיאות כוונה, כיוון שהוא אינו יכול לדעת את כוונתנו. הוא רק יכול לבצע את ההוראות שאנו נותנים לו. לפיכך, אין לו כל אפשרות לדעת אם התוצאה שהושגה היא התוצאה הרצויה לנו.

כדי לגלות שגיאות כוונה, עלינו לעקוב אחרי הביצוע ולנסות לגלות את ההוראה (או ההוראות) שהביאו לתוצאה השגויה.

במדעי המחשב, כל סוגי השגיאות נקראים **BUGS** והשלב של חיפוש גורם השגיאה ובידודו נקרא **ניפוי שגיאות** (**debugging**) .
לצורך ניפוי השגיאות, נוכל להיעזר ב**סימולציה** - ביצוע מדויק ושיטתי של התכנית כפי שהקומפיילר אמור לבצע אותה, תוך שימת דגש על כל אחת מהפעולות המתבצעות.
סימולציה זו נקראת: **טבלת מעקב.**

תרגילים

29. מטרת התכנית שלפניך היא לחשב את הממוצע החשבוני של ארבעה מספרים.
ממוצע חשבוני הינו סכום האיברים מחולק במספר האברים.
קלט: 4 מספרים שלמים. פלט: ממוצע חשבוני.
בתכנית שלפניך נפלו שגיאות מסוג: תחביר, ביצוע ושגיאות לוגיות.
תקן את השגיאות בתכנית בעזרת הטבלה שלמטה. (התייחס למספרי השורות)

```
1  public class Class1 {
2      public static void main(String[] args){
3          a, c, d  int ;
4          double  average
5          Scanner input = new Scanner (System.in);
6          System.out.println ("please enter 2 numbers  ) ;
7          a = input.nextint();
8          b = nextInt();
9          avg =  a + b/ 2 ;
10         println ("arithmetic average is " + average) ;
11     }
12 }
```

מספר שורה	תיאור השגיאה	התיקון

30. מטרת התכנית שלפניך היא לקלוט מספר שלם וחיובי תלת ספרתי, (הנח כי הקלט תקין) ולהדפיס את סכום הספרות ואת המספר בסדר ספרות הפוך.
בתכנית שלפניך נפלו שגיאות: תחביר, ביצוע ושגיאות לוגיות.
תקן את השגיאות בתכנית בעזרת הטבלה שלמטה. (התייחס למספרי השורות)

```
1  import java.util.Scanner;
2  public class Class1 {
3      public static void main(String[] args){
4          double num  ;
5          int  hundreds, tens, units, sumDigits  ;
6          Scanner input = new Scanner (System.in);
7          System.out.println ("please enter a 3 digits number ")
8          num = input.double();
9          hundreds  = num % 100 ;
10         tens  = num % 100 / 10 ;
11         units = num / 10 ;
12         sumDigits = hundreds + tens + units ;
13         System.out.println ("The new num is: " + units + tens + hundreds) ;
14         System.out.println ("Sum of digits is: + sum") ;
15     }
16 }
```

מספר שורה	תיאור השגיאה	התיקון