מרשם התושבים

שאלת בגרות 2001

במרשם התושבים של העיר "שמחה" שומרים את מספר המשפחות שיש להן 1, 2, 3, ... k ילדים, ואת מספר המשפחות שבהן יש רְצָפִים מסוימים של סֵדר הלידה של בנים ובנות.

לרצף כלשהו של סדר לידת ילדים שאינו קיים בעיר "שמחה" אין ייצוג במרשם התושבים.

<u>לדוגמה</u>: יש 80 משפחות שיש להן 3 ילדים, ומביניהן:

22 משפחות, שסדר לידת הילדים בהן הוא: בן, בן, בת

13 משפחות, שסדר לידת הילדים בהן הוא: בן, בת, בן

10 משפחות, שסדר לידת הילדים בהן הוא: בת, בת בת

35 משפחות, שסדר לידת הילדים בהן הוא: בן, בת, בת

לאחר לידת כל ילד מעדכנים את מרשם התושבים בעיר.

א. הצע ייצוב מתאים למרשם התושבים בעיר שמחה.
בתשובתך, הסבר את הייצוג והסבר איך הייצוג מתאים להראות את מספר המשפחות שיש להן רצף ילדים מסוים, ואיך יש לעדכן את המרשם לאחר לידת ילד.

לפניך שתי פעולות:

(i) **הוסף-דור** (מין הילוד, מקום-במבנה-הנתונים) **addGeneration**
הפעולה מקבלת את מרשם התושבים ואת מין הילוד, ומוסיפה דור חדש במקרה שלא קיים רצף כזה.
למשל, אם נולדה בת למשפחה שלה 3 ילדים: בן, בן, בת, ובמרשם התושבים לא קיים רצף בן, בן, בת, בת, יצור רצף כזה ויעדכן את מספר המשפחות בהתאם.

(ii) **עדכן-מרשם** (מין, רצף, מרשם) **update**
פעולה המקבלת את מרשם התושבים, רשימה המייצגת את רצף הילדים במשפחה לפני הלידה, ואת מין הילוד, ומעדכנת את מרשם התושבים בהתאם.

ב. ממש את הפעולה הוסף-דור לפי הייצוג שבחרת בסעיף א'.

ג. ממש את הפעולה עדכן-מרשם לפי הייצוג שבחרת בסעיף א'

ד. מהי סיבוכיות הפעולה עדכן-מרשם ? נמק

תרגול

ייצוג:

עץ בינארי. לידת בן תביא לְפנְיָה לענף הימני, ולידת בת לענף שמאלי.

ערך הצומת = מספר המשפחות שרצף לידת הילדים מתאים למסלול משורש העץ ועד הצומת.

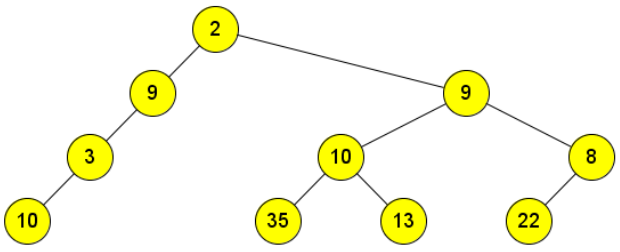
לדוגמה: ערך השורש מייצג את מספר המשפחות ללא ילדים (0 ילדים).

ערך הצמתים ברמה 1, מיצגים בצד שמאל - מספר המשפחות עם ילד אחד - בת,

בצד ימין - מספר המשפחות עם ילד אחד - בן.

שלב 1: בניית עץ בינארי שיתאים לנתוני השאלה (עבור 3 ילדים)

```
//--- (מבחינת 3 ילדים) פעולה הבונה את המרשם כך שיתאים ---
//---                                לתאור המרשם בשאלה ---
public static BinNode<Integer> buildMirsham ()
{
    //--- 2 בניית הצמתים ברמה ---
            BinNode<Integer> t1 = new BinNode<Integer>(3);

            BinNode<Integer> t3 = new BinNode<Integer>(10);
            BinNode<Integer> t4 = new BinNode<Integer>(8);

            //--- 1 בניית הצמתים ברמה ---
            BinNode<Integer> m1 = new BinNode<Integer> (t1, 9, null);
            BinNode<Integer> m2 = new BinNode<Integer> (t3, 9, t4);

            //--- 0 בניית שורש העץ - רמה ---
            BinNode<Integer> m = new BinNode<Integer> (m1, 2, m2);

            //--- 3 הוספת הצמתים ברמה ---
            t1.setLeft(new BinNode<Integer>(10));
            t3.setLeft(new BinNode<Integer>(35));
            t3.setRight(new BinNode<Integer>(13));
            t4.setLeft(new BinNode<Integer>(22));

            return m;
}

public static void main(String[] args)
{
    BinNode<Integer> m = buildMirsham ();
    TreeUtils.showTree(m, "מרשם התושבים");
```



שלב 2: עדכון העץ - הוספת ילוד למשפחה.

יתכנו שני מקרים:

- הוספת ילד אינה משנה את מבנה העץ:
  דוגמה: למשפחה עם 2 ילדים: בן, בת נולד בן חדש.
  קיים מסלול בן, בת, בן ולכן פעולת העדכון תפחית 1 ממספר המשפחות שלהן בן, בת
  ותוסיף 1 למספר המשפחות שלהן בן, בת, בן
  הפעולה update (m, lst, gender)

```
//--- פעולה המקבלת את מרשם התושבים, רשימה המתארת את רצף הילדים ---
//--- במשפחה לפני הלידה ואת מין הילוד, ומעדכנת את המרשם ---
public static void update (BinNode<Integer> m, Node<String> lst, String gender)
{
    boolean valid = true;
    Node<String>pos = lst;
    while (pos != null && valid)
    {
        if (m == null)  //  קיים רצף שאין לו ייצוג
        {
            System.out.println("Error!");
            valid = false;
        }
        else
        {
            if (pos.getValue().equals("girl") && m.getLeft() != null)
                m = m.getLeft();
            else
                if (pos.getValue().equals("boy") && m.getRight() != null)
                    m = m.getRight();
            pos = pos.getNext();
        }
    }

    //--- נמצא המקום, הוספת הילוד למרשם ---
    if (valid)
    {
        if (gender.equals("girl"))
        {
            if (m.hasLeft())
            {
                update (m.getLeft(), +1);
                update (m, -1);
            }
            else
                addGenetation(m, gender);
        }
        else
        {
            if (m.hasRight())
            {
                update (m.getRight(), +1);
                update (m, -1);
            }
            else
                addGenetation(m, gender);
        }
    }
}
```

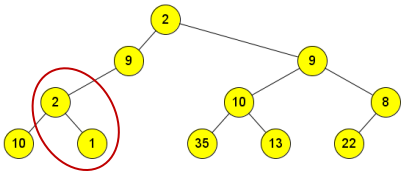
יעילות הפעולה **עדכון מרשם** היא $O(\log_2 n)$

n מיצג את מספר הצמתים בעץ.
רצף הילדים במשפחה אורכו לכל היותר $\log_2 n$.

הפעולה עוברת פעם אחת על מסלול התואם את רצף הילדים במשפחה.

פעולה העדכון או הוספת צומת לעץ נעשית ב- $O(1)$

```
//--- עדכון ערך הצומת בעץ ---
public static void update(BinNode<Integer> t, int k)
{
    t.setValue(t.getValue() + k);
}
```



- הוספת ילד משנה את מבנה העץ:

דוגמה: למשפחה שלה 2 בנות אחת נולד בן.

בעץ שבדוגמה לא קיים מסלול בת, בת, בן, לכן נוסיף צומת עם הערך 1 כבן ימני של "בת" ונפחית 1 מהצומת שמציג את מספר המשפחות שלהן ילד אחד מסוג בת.

הפעולה addGeneration (t, gender)

```
//--- פעולה המקבלת צומת בעץ ואת מין הילוד ---
//--- (לצומת אין בן המתאים למין הילוד) ---
//--- ומוסיפה צומת חדש לפי מין הילוד, תוך שמירה על מבנה הנתונים ---

public static void addGenetation (BinNode<Integer> t, String gender)
{
    if (gender.equals("girl") && t.getLeft() == null)
        t.setLeft(new BinNode<Integer>(1));
    else
        if (t.getRight() == null)
            t.setRight(new BinNode<Integer>(1));

    //--- עדכון ערך הצומת (שהתאים למבנה הישן של המשפחה) ---
    update (t, -1);
}
```

הערה: אם למשפחה נולדו תאומים, נבצע פעמיים את פעולת update לפי סדר הלידה של התאומים.

```java
static Random rnd = new Random();

//--- פעולה המחזירה רשימה המכילה את רצף ילדים במשפחה בת n נפשות ---
public static Node<String> getFamilyStructure (int n)
{
    String [] arr = {"boy", "girl"};
    Node<String> lst = null;
    Node<String>pos = null;
    int k;

    for (int i = 0 ; i < n ; i++)
    {
        k = rnd.nextInt(2);
        if (pos == null)
        {
            lst = new Node<String> (arr[k]);
            pos = lst;
        }
        else
        {
            pos.setNext(new Node<String> (arr[k]));
            pos = pos.getNext();
        }
    }
    return lst;
}
```

```
public static void main(String[] args)
{
    BinNode<Integer> m = buildMirsham ();
    TreeUtils.showTree(m, "מרשם התושבים");

    Node<String> lst = getFamilyStructure(2);
    update (m, lst, "boy");
    TreeUtils.showTree(m, "מרשם התושבים לאחר עדכון");

    lst = getFamilyStructure(3);
    update (m, lst, "girl");
    TreeUtils.showTree(m, "מרשם התושבים לאחר עדכון נוסף");
}
```