© איילת ויץ

חלק ב'

ענו על שתיים מבין השאלות 5-8 (ערך כל שאלה – 18 נקודות).

שאלה 5

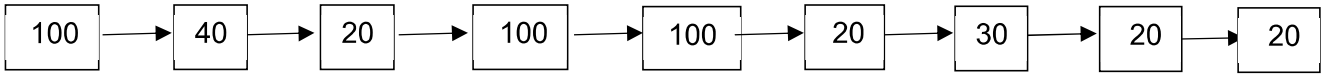
נתונה רשימה של מספרים שלמים. כל איבר (מספר) ברשימה יכול להופיע פעם אחת או יותר.

רשימת תדירויות (Frequency List) היא רשימה של מספרים שלמים שבה כל מספר מופיע פעם אחת ולאחריו מספר הפעמים שהופיע ברשימה המקורית.

(7 נק') א. כתבו פעולה המקבלת רשימה של מספרים שלמים (הפנייה לחוליה ראשונה של הרשימה), ברשימה יש לפחות ערך אחד. הפעולה תשנה את הרשימה להיות **רשימת תדירויות.**

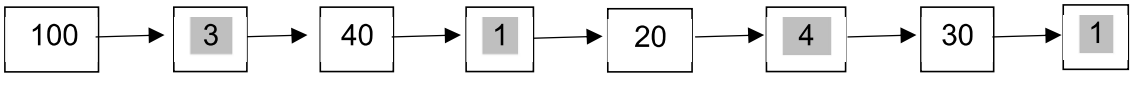
שימו לב: הפעולה תעדכן את הרשימה שהתקבלה כך שכל ערך יופיע פעם אחת ולאחריו מספר הפעמיים שהופיע ברשימה המקורית.
יש למחוק ערכים שחוזרים על עצמם חוץ מהמופע הראשון.
אין להשתמש במבני נתונים נוספים!!!

לדוגמה:
הרשימה המקורית:



- מספר 100 מופיע שלוש פעמים.
- מספר 40 מופיע פעם אחת.
- מספר 20 מופיע ארבע פעמים.
- מספר 30 מופיע פעם אחת.

הרשימה אחרי העדכון (רשימת תדירויות):



כותרת הפעולה:

```
public static void buildFreqList(Node<Integer> chain)
```

(7 נק') ב. כתבו פעולה המקבלת הפניה לחוליה ראשונה של שרשרת חוליות ומחזירה את המספר המופיע הכי הרבה פעמים בשרשרת. בעבור הרשימה הנתונה בסעיף א' הפעולה תחזיר 20.

כותרת הפעולה:

```
public static int mostPopularNumber(Node<Integer> chain)
```

חובה להשתמש בפעולה שכתבתם בסעיף א'.

(4 נק') ג. מהי סיבוכיות הפעולות `buildFreqList` ו-`mostPopularNumber` מהסעיפים א' ו-ב'?

הסבירו את תשובתכם.

5. נתונה המחלקה **NumCount** – מספר ערכים, ולה שתי תכונות:

- num – ערך מספרי, מטיפוס שלם.
- count – מספר המופעים של הערך (num) , מטיפוס שלם. המספר גדול או שווה ל- 0 .

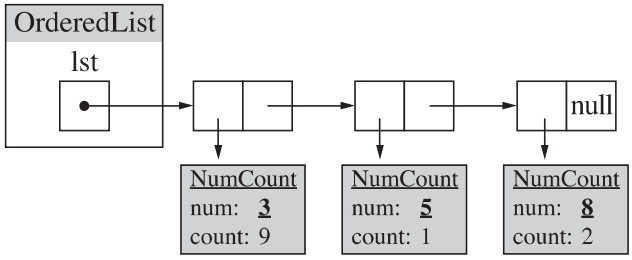
הניחו שקיימות פעולות get/Get ו- set/Set לכל אחת מן התכונות במחלקה, ופעולה בונה המקבלת ערכים עבור תכונות המחלקה.

נתונה המחלקה **OrderedList** – שרשרת ממוינת, ולה תכונה אחת:

- lst – מצביע על ראש של שרשרת חוליות מטיפוס **NumCount** .

שרשרת החוליות ממוינת לפי סדר עולה של ערך התכונה – num . ערך התכונה num שונה בכל חוליה.

<u>דוגמה</u>: השרשרת שלפניכם מקיימת את תנאי המחלקה (השרשרת ממוינת בסדר עולה לפי ערך התכונה num , וערך התכונה num שונה בכל חוליה).



א. **(1)** ממשו במחלקה OrderedList את הפעולה הפנימית שלפניכם:

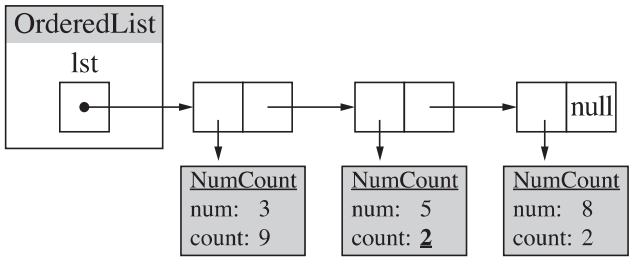
Java – public void insertNum (int x)

C# – public void InsertNum (int x)

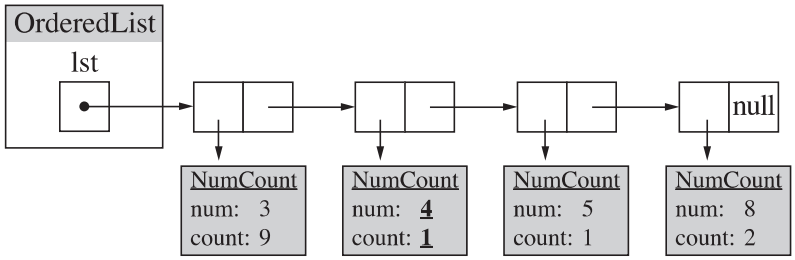
הפעולה מוסיפה את הערך של x לשרשרת באופן שלפניכם:

- אם קיימת בשרשרת חוליה שהתכונה num שלה שווה ל- x , הפעולה תגדיל ב- 1 את התכונה count (כמות המופעים) באותה החוליה.
- אם השרשרת ריקה או שלא קיימת בשרשרת חוליה שהתכונה num שלה שווה ל- x , הפעולה תכניס חוליה חדשה, שבה התכונה num תהיה שווה ל- x והתכונה count תהיה שווה ל- 1 , במיקום השומר את הסדר העולה של השרשרת.

<u>דוגמה</u>: עבור השרשרת המוצגת לעיל ו- $x = 5$, בתום הפעולה תיראה השרשרת כך:



<u>דוגמה נוספת</u>: עבור אותה השרשרת המוצגת לעיל (בדוגמה הראשונה) ו- $x = 4$, בתום הפעולה תיראה השרשרת כך:

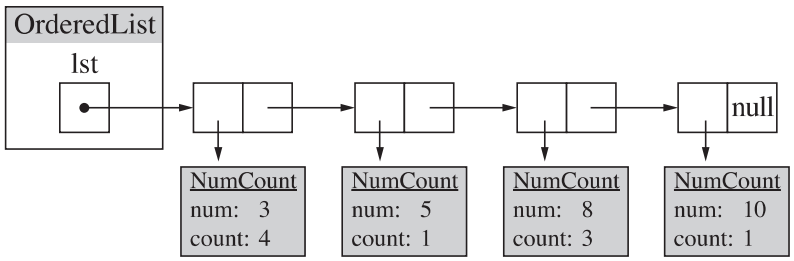


(2) מהי סיבוכיות זמן הריצה של הפעולה שכתבתם בסעיף א(1)? נמקו את תשובתכם.

(שימו לב: המשך השאלה בעמוד הבא.)

ב. "ערך המופע ה־ n " הוא הערך שמופיע בַּמָקום ה־ n לפי הסדר מתחילת השרשרת (בשקלול כמות המופעים – count של כל ערך).

לדוגמה: עבור השרשרת שלפניכם ו־ $n = 7$ הפעולה תחזיר את הערך 8 .



הסבר: סדר הערכים של השרשרת ברצף, בהתאם לכמות המופעים שלהם, הוא: 10 , 8 , **8** , 8 , 5 , 3 , 3 , 3 , 3 , 3 .

$n = 7$ ובמקום השביעי ברצף מופיע הערך 8 . לכן הפעולה תחזיר את הערך 8 .

ממשו במחלקה OrderedList את הפעולה הפנימית שלפניכם:

Java – public int valueN (int n)

C# – public int ValueN (int n)

הפעולה מקבלת את המספר n , ומחזירה את "ערך המופע ה־ n ".

הניחו ש"ערך המופע ה־ n " קיים בשרשרת.

5. נתונה המחלקה **Patient** – חולה בחדר מיון, ולה שתי תכונות:

- id – מספר הזהות של החולה, מטיפוס שלם
- priority – רמת הדחיפות של הטיפול בחולה. רמת הדחיפות מיוצגת במספר מטיפוס שלם בין 1 ל־ 10 . ככל שהמספר גבוה יותר, רמת הדחיפות גבוהה יותר.

הניחו שלתכונות המחלקה יש פעולות get/Get ו־ set/Set .

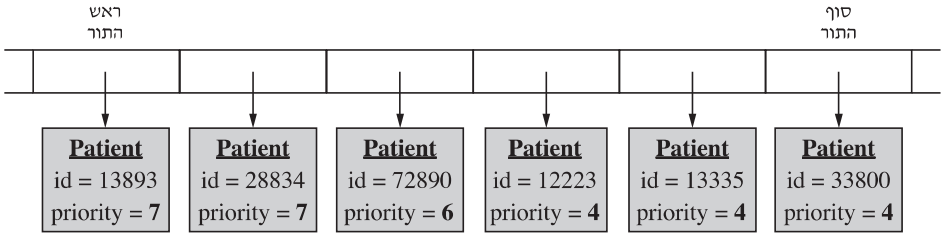
סדר הטיפול בחולים בחדר המיון מתנהל באופן שלהלן:

ככל שרמת הדחיפות של הטיפול גבוהה יותר, החולה מטופל מוקדם יותר. כאשר יש יותר מחולה אחד באותה רמת דחיפות, החולה שהגיע קודם לחדר מיון מטופל מוקדם יותר.

כדי לשמור על סדר הטיפול נבנתה המחלקה **PriorQueue** – תור עדיפויות, ולה תכונה אחת:

- q – הפניה לתור, מטיפוס Patient

דוגמה לתור q :



הניחו שהחולים שרמת הדחיפות שלהם זהה מוצגים בדוגמה לפי סדר הגעתם לחדר המיון.

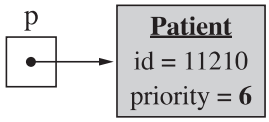
א. ממשו את הפעולה שלפניכם השייכת למשק המחלקה PriorQueue :

Java – public void priorityInsert (Patient p)

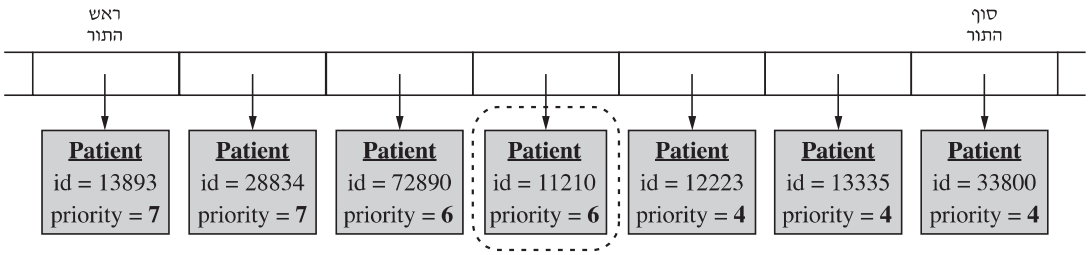
C# – public void PriorityInsert (Patient p)

הפעולה מקבלת חולה חדש – p ומכניסה אותו לתור q בהתאם לכללים של חדר המיון הכתובים לעיל.

לדוגמה: עבור התור המוצג לעיל והעצם שלפניכם:



התור ייראה כך לאחר ההכנסה:



(שימו לב: המשך השאלה בעמוד הבא.)

ב. מדי פעם רמת הדחיפות של חולה מסוים משתנה במהלך שהותו בחדר המיון.

ממשו את הפעולה שלפניכם השייכת לממשק המחלקה PriorQueue :

Java – public void update (int id, int pri)

C# – public void Update (int id, int pri)

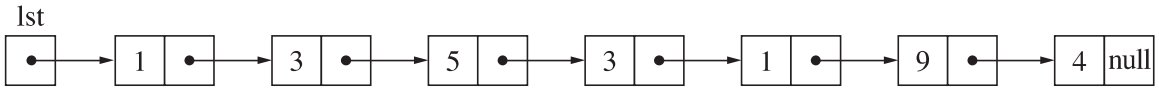
הפעולה מקבלת מספר זהות של חולה הנמצא בתור – id , ומספר – pri המייצג את רמת הדחיפות המעודכנת שלו.
הפעולה תעדכן את התכונה priority של החולה ותמקם אותו בתור לטיפול במקום המתאים לו (בהתאם לרמת הדחיפות המעודכנת – pri). אם בתור כבר יש חולים אחרים באותה רמת דחיפות – pri , החולה הנוכחי – id יוצב אחריהם כאילו הגיע אחריהם לחדר המיון.

לפותרים בשפת C#

א. נתונה הפעולה What :

```
public static Node<int> What (Node<int>lst, int x)
{
    if (lst == null)
        return null;
    Node<int> temp = What (lst.GetNext(),x);
    if (lst.GetValue() == x)
        return temp;
    lst.SetNext (temp);
    return lst;
}
```

נתונה שרשרת חוליות – lst מטיפוס שלם:



(1) עקבו אחר הפעולה What (lst, 1) , והציגו את השרשרת שהפעולה מחזירה.

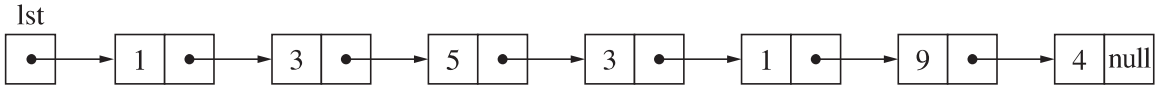
(2) מה עושה הפעולה What ? הסבירו את תשובתכם.

(3) מהי סיבוכיות הפעולה What ? נמקו את תשובתכם.

ב. נתונה הפעולה Guess :

```
public static void Guess (Node<int>lst)
{
    if (lst != null)  {
        Node<int> temp = What (lst.GetNext(), lst.GetValue());
        lst.SetNext (temp);
        Guess (lst.GetNext());
    }
}
```

נתונה שרשרת חוליות – lst מטיפוס שלם:



(1) עקבו אחר הפעולה Guess (lst) , והציגו את השרשרת lst בסיום הפעולה.

בסעיף זה אין צורך לעקוב אחר הפעולה What .

(2) מה עושה הפעולה Guess ? הסבירו את תשובתכם.

(3) מהי סיבוכיות הפעולה Guess ? נמקו את תשובתכם.

/ המשך בעמוד 10

7. נתונה המחלקה **BusStation** – תחנת אוטובוס, ולה שלוש תכונות:

- num – מספר התחנה, מטיפוס שלם
- arr – מערך מטיפוס שלם בגודל 10 , המכיל את מספרי קווי האוטובוס שעוצרים בתחנה. בתחנה עוצרים עד 10 קווים, והם נשמרים ברצף מתחילת המערך.
- amount – כמות קווי האוטובוס שעוצרים בתחנה בפועל, מטיפוס שלם. בתחנה עוצר קו אוטובוס אחד לפחות.

הניחו שלתכונות המחלקה יש פעולות get/Get ו- set/Set .

א. ממשו את הפעולה שלפניכם השייכת למחלקה BusStation :

Java – public boolean isStopping (int n)

C# – public bool IsStopping (int n)

הפעולה מקבלת מספר קו אוטובוס – n מטיפוס שלם. הפעולה מחזירה true אם קו האוטובוס עוצר בתחנה, ואחרת מחזירה false .

ב. נתון מערך arr מטיפוס BusStation , ובו כל תחנות האוטובוס בעיר מסוימת. ידוע שכמה מקווי האוטובוס עוצרים בכל אחת מן התחנות בעיר, ושאר הקווים עוצרים רק בחלק מן התחנות.

כתבו פעולה חיצונית ששמה allStations בשפת Java או AllStations בשפת C# , המקבלת את המערך arr .

הפעולה מחזירה שרשרת חוליות מטיפוס שלם שבכל חוליה מופיע אחד ממספרי הקווים שעוצרים ב**כל** התחנות בעיר (כל קו כזה יופיע פעם אחת בלבד בשרשרת).

הערה: אין חשיבות לסדר הקווים בשרשרת.

פרק שני (50 נקודות)

שימו לב: בכל שאלה שנדרש בה מימוש אפשר להשתמש בפעולות של המחלקות: תור, מחסנית, עץ בינרי וחוליה, בלי לממש אותן. אם משתמשים בפעולות נוספות, יש לממש אותן.

ענו על שתיים מן השאלות 4–7 (לכל שאלה – 25 נקודות).

4. נתונה המחלקה **Range** – טווח, ולה שתי תכונות:

- low – מספר מטיפוס שלם
- high – מספר מטיפוס שלם

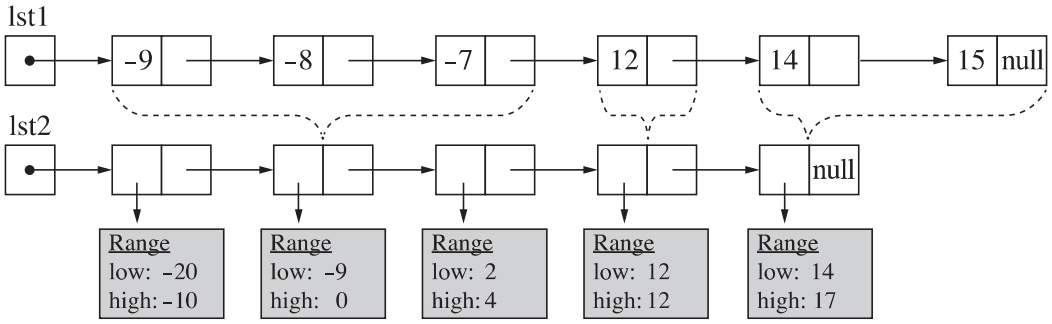
המספר high גדול או שווה ל- low ($high \geq low$).

הניחו שיש פעולות get/Get ו- set/Set בעבור תכונות המחלקה.

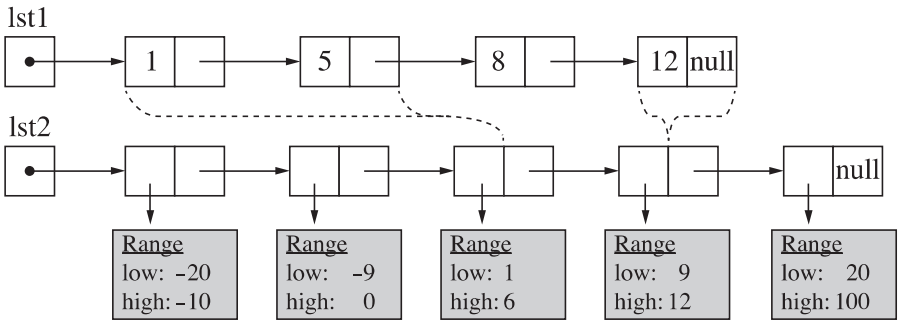
מספר כלשהו, x, "מוכל" בעצם מטיפוס Range אם הוא נמצא בטווח המספרים שבין low ובין high ($high \geq x \geq low$).

שרשרת חוליות – lst1 מטיפוס שלם "מוכלת" בשרשרת חוליות – lst2 מטיפוס Range אם בעבור כל מספר בשרשרת lst1 קיימת חוליה בשרשרת lst2 המכילה אותו.

דוגמה לשרשרת lst1 המוכלת בשרשרת lst2 :



דוגמה לשרשרת lst1 ש**אינה** מוכלת בשרשרת lst2 :



הסבר: המספר 8 שבשרשרת lst1 אינו "מוכל" בשום חוליה בשרשרת lst2 .

(שימו לב: המשך השאלה בעמוד הבא.)

/ המשך בעמוד 7 /

ממשו את הפעולה החיצונית שלהלן:

Java – public static boolean isIncluded (Node<Integer> lst1, Node<Range> lst2)

C# – public static bool IsIncluded (Node<int> lst1, Node<Range> lst2)

הפעולה מחזירה true אם lst1 "מוכלת" ב־ lst2, אחרת היא מחזירה false. הפעולה חייבת לעבוד בסיבוכיות זמן ריצה של O(N).

הערה: N הוא אורך השרשרת הארוכה יותר מבין שתי השרשראות.

הנחות:

- lst1 ו־ lst2 אינם null.
- בשרשרת lst2 כל העצמים מטיפוס Range אינם null.
- **השרשרת lst1 ממוינת בסדר עולה.**
- **השרשרת lst2 ממוינת בסדר עולה**, כלומר, ערך ה־ high של כל חוליה קטן מערך ה־ low של החוליה הבאה אחריה בשרשרת (כפי שמופיע בדוגמאות בעמוד הקודם).

/המשך בעמוד 8/