מועד הבחינה:
קיץ תשפ"ב – 2022 – מועד ב'
מספר השאלון: 97105
נספח ממשקים לבחינה JAVA
נספח ממשקים לבחינה #C
נספח תרגום לערבית

מבני נתונים ותכנות מונחה עצמים

הנדסאים וטכנאים – הנדסת תוכנה

הנחיות לבחינה

א. משך הבחינה: ארבע שעות וחצי.

ב. מבנה השאלון ומפתח ההערכה:

בשאלון זה שני מבחנים, <u>עליכם לענות על מבחן אחד בלבד בהתאם למוסד הלימודים</u>:

מבחן ב- Java (עמוד 2)

מבחן ב- #C (עמוד 14)

בכל מבחן 11 שאלות.

חלק א' – 45 נקודות

שאלות 1-4: יש לענות על **שלוש** שאלות בלבד. ערך כל שאלה 15 נקודות.

חלק ב' – 30 נקודות

שאלות 5-8: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 15 נקודות.

חלק ג' – 25 נקודות

שאלות 9-11: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 12 נקודות.

נקודה אחת תינתן על הערכה.

בסך הכול: 100 נקודות.

ג. חומר עזר מותר לשימוש:

1. מחשבון (אין להשתמש במחשב כף יד או במחשבון עם תקשורת חיצונית).
2. קלסר **<u>אחד בלבד</u>** עם **<u>חומר ההרצאות</u>**. אין להוציא דפים מהקלסר.

אין לצרף ספרים או חוברות עם פתרונות.

ד. הוראות כלליות:

1. יש לקרוא בעיון את ההנחיות בדף השער ואת כל שאלות הבחינה, ולוודא שהן מובנות.
2. את התשובות יש לכתוב בצורה מסודרת, בכתב יד ברור ונקי (גם בכך תלויה הערכת הבחינה).
3. יש להשאיר את העמוד הראשון במחברת הבחינה ריק. בסיום המבחן יש לרשום בעמוד זה את מספרי התשובות לבדיקה. התשובות ייבדקו לפי סדר כתיבתן בעמוד זה.
   לא ייבדקו תשובות עודפות.
4. יש לכתוב את התשובות במחברת הבחינה **בעט בלבד**, בכתב יד ברור.
5. יש להתחיל כל תשובה בעמוד חדש ולציין את מספר השאלה ואת הסעיף.
   אין צורך להעתיק את השאלה עצמה.
6. טיוטה יש לכתוב במחברת הבחינה בלבד. יש לרשום את המילה "טיוטה" בראש העמוד ולהעביר עליו קו כדי שלא ייבדק.
7. יש להציג פתרון מלא ומנומק, כולל חישובים לפי הצורך. הצגת תשובה סופית ללא שלבי הפתרון לא תזכה בניקוד.
8. יש להסביר בפירוט כל תוכנית שנכתבה, **תוכנית ללא הסבר מפורט לא תזכה בניקוד**.
9. אם לדעתכם חסר בשאלה נתון, יש לציין זאת ולהוסיף נתון מתאים שיאפשר לכם להמשיך בפתרון השאלה, נמקו את בחירתכם.

חל איסור מוחלט להוציא שאלון או מחברת בחינה מחדר הבחינה!

בהצלחה!

בשאלון זה 25 עמודי בחינה ו- 6 עמודי נספחים.

מבחן ב- JAVA

חלק א'

ענו על <u>שלוש</u> מבין השאלות 1-4 (ערך כל שאלה – 15 נקודות).

שאלה 1

מחסנית "שוות סכומים", היא מחסנית של מספרים שלמים המכילה מספר אי זוגי של איברים.

כך שכל שני איברים קיצוניים (ראשון + אחרון, שני + לפני אחרון וכו') שווים בסכומם לאיבר האמצעי מחסנית.

לדוגמה:

המחסנית הבאה היא "מחסנית שווה סכומים"

18	3	15	13	4	**25**	21	12	10	22	7

האיבר האמצעי במחסנית הוא 25. וכל זוג איברים קיצונים שווה ל- 25.

25=18+7 (האיבר הראשון והאיבר האחרון).

25=3+22 (האיבר השני והאיבר שלפני אחרון).

25=15+10

25=13+12

25=4+21

(12 נק') א. כתבו פעולה בשם `equalSums` המקבלת מחסנית של מספרים שלמים. הפעולה מחזירה true אם המחסנית היא מחסנית "שווה סכומים", ואם היא אינה "שוות סכומים", הפעולה תחזיר false.

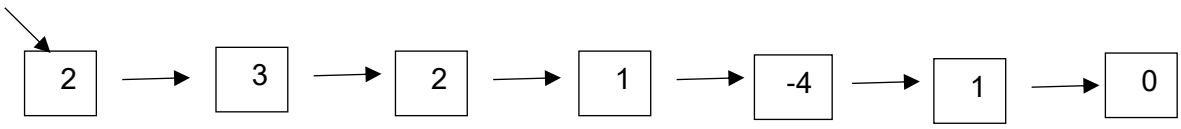
(3 נק') ב. מהי הסיבוכיות של הפעולה שכתבתם בסעיף א'? הסבירו את תשובתכם.

שאלה 2

נגדיר: "**רשימת מקטעים**" כשרשרת חוליות של מספרים שלמים ריקה או כשרשרת חוליות אשר כל אחת מהחוליות שלה מכילה ערך **קטן או שווה** למספר החוליות העוקבות לה (נמצאות אחריה בשרשרת).

לדוגמה:

השרשרת הבאה היא "רשימת מקטעים":



הערך של האיבר הראשון בשרשרת **(2)** הוא קטן מ**מספר** האיברים שאחריו (6).

הערך של האיבר החמישי בשרשרת **(-4)** הוא קטן מ**מספר** האיברים שאחריו (2).

(5 נק') א. כתבו פעולה חיצונית בשם numNodesFollowing המקבלת הפנייה לחוליה בשרשרת כלשהי ומחזירה את מספר החוליות העוקבות לה.

(10 נק') ב. כתבו פעולה חיצונית בשם isSection המקבלת הפניה לשרשרת חוליות מטיפוס שלם. הפעולה מחזירה true אם השרשרת היא "רשימת מקטעים", ואם לא, הפעולה תחזיר false.

שאלה 3

נתונות ארבע המחלקות הבאות:

```
public  class A
{
    protected int x;
    public A()
    {
        this.x = 1;
    }
    public A(int x)
    {
        this.x = x;
    }
    public void change()
    {
        this.x = this.x * 2;
    }
    public String toString()
    {
        return "X=" + this.x;
    }
}

public  class B extends A
{
    protected int y;

    public B(int y)
    {
        this.y = -y;
    }
    public B(int x,int y)
    {
        super(x);
        this.y = -y;
    }
    public String toString()
    {
        return super.toString() + " Y=" + this.y;
    }
}
```

```
public  class C extends B
{
    private int z;
    public C(int x,int y,int z)
    {
        super(x,y);
        this.z = z;
    }
    public void change()
    {
        this.x = 3 * this.x;
        this.y = 3 * this.y;
        this.z = 3 * this.z;
    }
    public String toString()
    {
        return super.toString() + " Z=" + this.z;
    }
}

public class TestABC
{
    public   static void main(String[] args)
    {
        A[] data = new A[5];
        data [0] = new A (5);
        data [1] = new B(5,10);
        data [2] = new C(5,10,15);
        data [3] = data [0];
        data [4] = new B(5);

        for (int i = 0; i < data.length; i++)
            System.out.println(data[i]);

        for(int i=0;i< data.length;i++)
            data [i].change();
        for (int i = 0; i < data.length; i++)
            System.out.println(data[i]);
    }
}
```

עקבו אחרי ביצוע פעולה הראשית (main) של המחלקה TestABC ורשמו מה יהיה הפלט של הפעולה הראשית.

חובה להציג עבור כל עצם שנוצר בפעולה את ערכי התכונות שלו.

שאלה 4

נתונות ארבע המחלקות הבאות:

```
public class B
{
    protected int num;
    public B( int num)
    {
        this.num = num;
    }
} // end of B

public class C extends B
{
    public C(int num)
    {
        super(num);
    }
    public boolean equals(Object other)
    {
        return ((other!=null) && (other instanceof C) &&
        (num == ((C) other).num));
    }
 } // end of C

public class D extends B
{
    public D( int num)
    {
        super(num+1);
    }

    public boolean equals(D other)
    {
        return ((other!=null) && (num == other.num));
    }
} // end of D
```

(4 נק׳) א. האם קיימת דריסה (overriding) או העמסה (overloading) של הפעולות?

הסבירו את תשובתכם.

נתונה הפעולה ראשית main במחלקה Program

```
public class Program
{
    public static void main(String[] args)
    {
        B b = new B(1);
        C c = new C(1);
        D d = new D(1);
        B b1 = new D(1);
        Object c1 = new C(1);
        Object d1 = new D(1);
        (******)
     }
}
```

(3 נק') ב. עקבו אחרי ביצועה וציינו מהם העצמים שנוצרו. עבור כל עצם יש להציג את ערך תכונת ה- num שלו.

(8 נק') ג. בכל אחד מסעיפים הבאים השורה (*****) תוחלף בשורת קוד משורות הקוד 1-4.

כתבו בעבור כל אחד מהסעיפים מה יהיה הפלט ואיזו מהפעולות equals תופעל.

שימו לב שאין קשר בין סעיפים!

```
1)   System.out.println(b1.equals(b));
2)   System.out.println(c1.equals(c));
3)   System.out.println(d1.equals(d));
4)   System.out.println(d.equals((D)d1));
```

חלק ב'

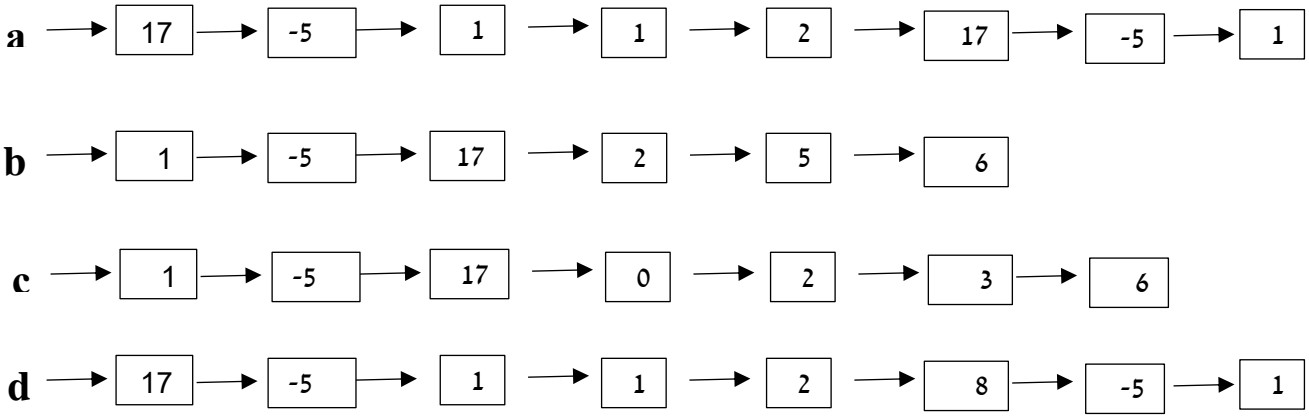
ענו על <u>שתיים</u> מבין השאלות 5-8 (ערך כל שאלה – 15 נקודות).

שאלה 5

נתונה הפעולה הבאה:

```
public static void what(Node<Integer>f, Node<Integer> s)
{
  Stack<Integer>st = new Stack<Integer>();
  while(s!=null)
  {
     st.push(s.getValue());
     s = s.getNext();
  }
  while(f!=null && !st.isEmpty() && (st.pop()==f.getValue()))
  {
     System.out.print(f.getNext()+", ");
     f = f.getNext();
  }
    System.out.println("STOP!");
}
```

נתונות ארבע השרשרות הבאות:



(6 נק') א. עקבו אחרי זימון הפעולה `what(a, b)` ורשמו מה יהיה הפלט של הזימון. <u>יש להציג את המעקב</u>!

(6 נק') ב. עקב זימון הפעולה what התקבל הפלט (משמאל לימין):

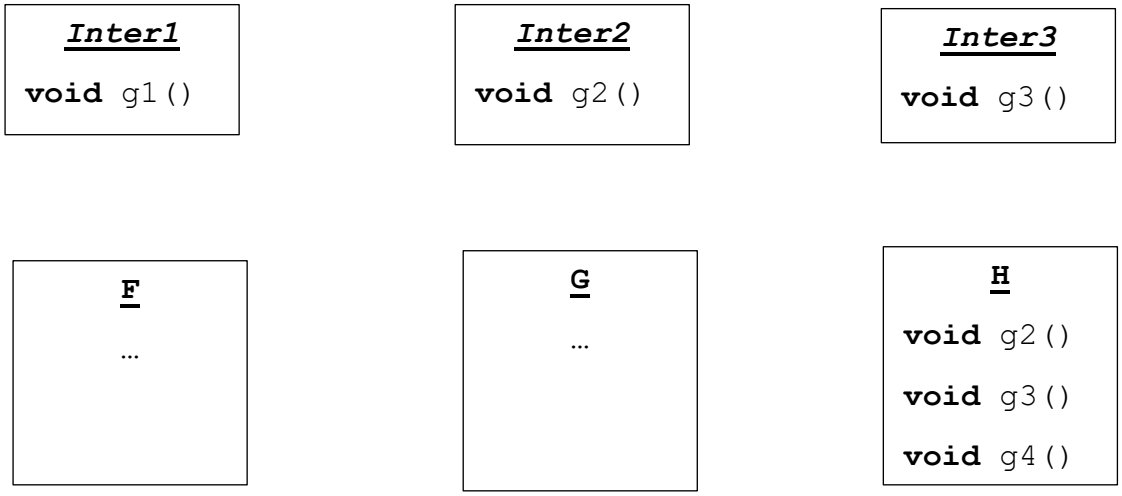
```
1, -5, 17, STOP!
```

אילו פרמטרים נשלחו לפעולה? יש לבחור זוג פרמטרים מתוך ארבע השרשרות הנתונות **(a, b, c, d)**.

(3 נק') ג. מה מבצעת הפעולה באופן כללי?

שאלה 6

לפניכם שלושה ממשקים ושלוש מחלקות:



נתון קטע מפעולה ראשית. הקטע עובד <u>ללא שגיאות הידור או שגיאות זמן ריצה</u>.

```
Inter1 x = new F();
G y = new G();
Inter3 z = new H();
x = y;
y = new H();
y.g2();
z.g3();
((H)z).g4();
z = new F();
```

(8 נק׳) א. השלימו את תרשים UML: הוספו כותרות הפעולות במחלקות F ו- G, ציירו קשרים בין המחלקות והממשקים.

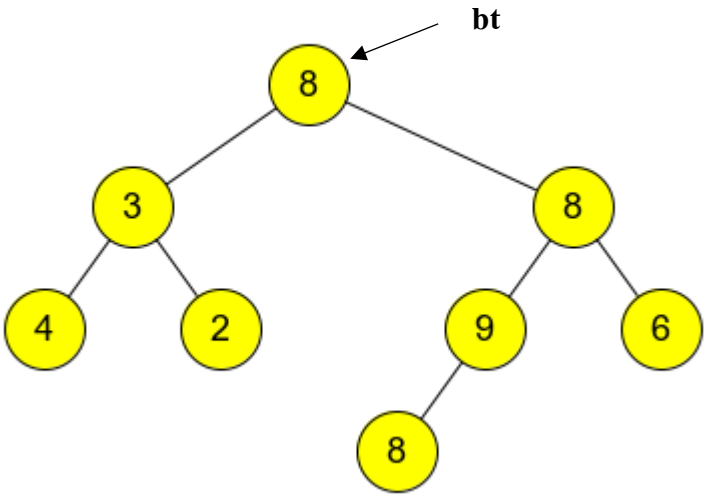
(7 נק׳) ב. כתבו פעולה המקבלת מערך עצמים מסוג Object. הפעולה מדפיסה את מספר האיברים שעבורם אפשר להפעיל את הפעולה **g1()** ואת מספר האיברים שעבורם <u>אי אפשר</u> להפעיל את הפעולה **g3()**.

שאלה 7

לפניכם ארבע פעולות:

```
public static int first(BinNode<Integer> bt){
    if(bt==null)
        return 0;
    return
       bt.getValue()+first(bt.getLeft())+first(bt.getRight());
}
public static int second(BinNode<Integer> bt){
    return first(bt.getLeft())+first(bt.getRight());
}
public static void third(BinNode<Integer> bt){
    if(bt!=null)
    {
      bt.setValue(second(bt));
      third(bt.getLeft());
      third(bt.getRight());
    }
}
```

נתון העץ הבינרי הבא:



(5 נק') א. מה תהיה תוצאת זימון הפעולה **second(bt)**? **חובה להראות את המעקב!**

(3 נק') ב. מה מחזירה הפעולה **second(BinNode<Integer> bt)** באופן כללי?

(5 נק') ג. מה תהיה תוצאת הזימון **third(bt)** לעץ הנתון **bt**?

חובה להראות את המעקב אחרי הפעולה **third**. אין צורך במערב אחרי הפעולה second.

(2 נק') ד. מה מבצעת הפעולה **third(BinNode<Integer> bt)** באופן כללי?

שאלה 8

נתונה המחלקה **MyNode** הבאה:

```
public class MyNode
{
    private int value;
    private int howManyBig;
    private MyNode next;
```

למחלקה הוגדר בנאי:

```
   public MyNode(int val)
    {
      this.value = val;
      this.howManyBig = 0;
      this.next = null;
    }
```

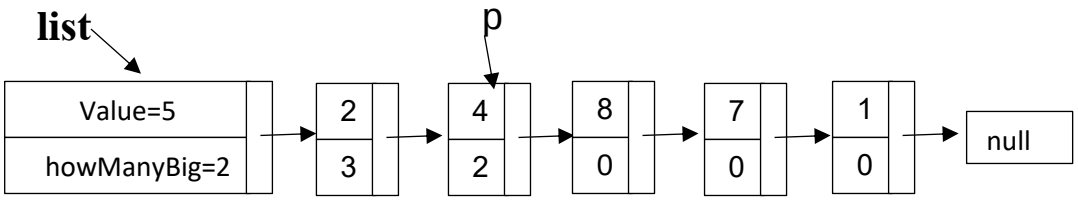
לכל תכונה במחלקה הוגדרו פעולות get/set

```
}
```

בעזרת המחלקה `MyNode` נבנית שרשרת חוליות לפי הכלל הבא:

כל חוליה היא מסוג `MyNode`. בכל איבר התכונה value מאחסנת מספר שלם, התכונה `howManyBig` מאחסנת מספר שלם ששווה ל**מספר** האיברים בשרשרת שנמצאים **אחרי** האיבר הנוכחי, שערך התכונה value שלהם גדול מערך `value` של האיבר הנוכחי. התכונה `next` מחזיקה מצביע על האיבר הבא.

לדוגמה:



P מצביע על האיבר שהתכונה value שלו שווה ל- 4, והתכונה `howManyBig` שווה ל- 2 כי יש שני איברים אחריו, שערכי ה- `value` שלהם גדולים מ- 4 (כלומר 8 ו- 7).

(12 נק') א. כתבו פעולה חיצונית המקבלת הפנייה לחוליה הראשונה של שרשרת החוליות מסוג `MyNode`, מספר שלם `val` ומספר שלם וחיובי position. כותרת הפעולה:

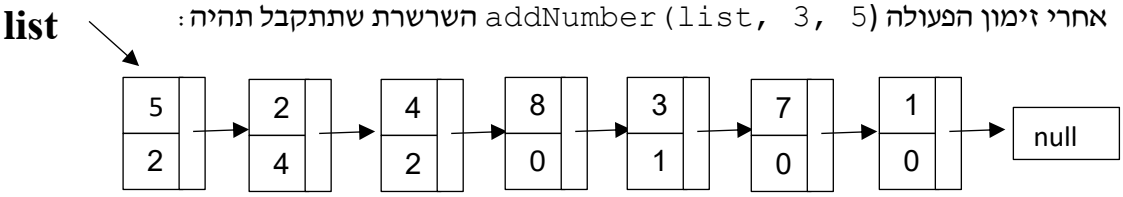
```
public static MyNode addNumber(MyNode list, int val, int position)
```

הפעולה צריכה להכניס איבר חדש לשרשרת במיקום `position` (המיקום של האיבר הראשון בשרשרת הוא 1). ערך התכונה `value` של האיבר החדש יהיה שווה לפרמטר `val`.

השרשרת שתתקבל צריכה לשמור על החוקים של השרשרת המקורית לגבי התכונה `howManyBig`.

לדוגמה:

אחרי זימון הפעולה `addNumber(list, 3, 5)` השרשרת שתתקבל תהיה:



(3 נק') ב. מהי סיבוכיות הפעולה? הסבירו את תשובתכם.

חלק ג'

ענו על <u>שתיים</u> מבין השאלות 9-11 (ערך כל שאלה – 12 נקודות).

שאלה 9

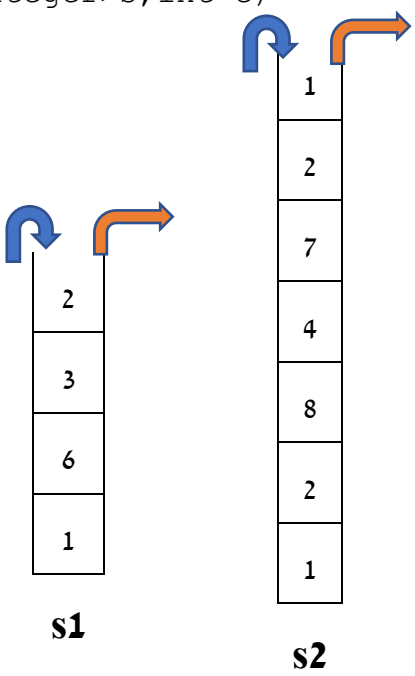
נתונות שתי הפעולות הרקורסיביות **secret** ו- **mystery**:

הפעולה **secret** מקבלת מחסנית לא ריקה של מספרים שלמים.

```
public static int secret (Stack<Integer> s)
{
   int x=s.pop();
   if(s.isEmpty())
     return x;
   int y = secret(s);
   s.push(x);
   return y;
}
```

הפעולה mystery מקבלת מחסנית של מספרים שלמים ומספר שלם וחיובי c.

```
 public static boolean mystery (Stack<Integer>s,int c)
 {
     if (c==0 ||  s.isEmpty())
          return true;
     int y=secret(s);
     if (s.isEmpty())
          return false;
     int x=s.pop();
     if (x==y)
          return mystery (s, c-1);
     return false;
}
```



כמו כן, נתונות המחסניות s1 ו- s2

(3 נק') א. עקבו אחר זימון הפעולה secret עבור המחסנית s1. מהו הערך שיוחזר מהפעולה?

יש להראות בכל שלב את הערכים של x, y ואת המחסנית s1.

(2 נק') ב. מהי מטרת הפעולה **secret**?

(5 נק') ג. עקבו אחר זימון הפעולה **mystery(s2, 2)** ורשמו מהו הערך שיוחזר מהפעולה.

יש להראות בכל שלב את הערכים של y, x, c, ואת המחסנית s. אין צורך לעקוב אחרי **secret**.

(2 נק') ד. מהי מטרת הפעולה **mystery**?

שאלה 10

<u>הערה</u>: בשאלה זאת הניחו שהפעולות get/set הוגדרו בעבור כל תכונה בכל אחת מהמחלקות.
אם הוספתם פעולות עזר נוספות, יש לציין את המחלקה שבה נמצאת פעולת העזר.

בחניון "לב העיר" קיימות מספר קומות. בכל קומה יש אזורים המזוהים לפי צבע. מספר האזורים אינו שווה בהכרח בכל קומה.

כל **אזור (Area)** מאופיין באמצעות:

- צבע.
- מספר מקומות פנויים כרגע.
- רשימת מספרי הרישוי של המכוניות שחונות בו כרגע (סדר המכוניות לא משנה).

כל **קומה (Floor)** מאופיינת על ידי מספר הקומה ואוסף האזורים שבה.

(3 נק') א. כתבו מחלקות עבור הטיפוסים: **אזור**– **Area** ו**קומה** – **Floor**.
לכל מחלקה כתבו כותרת מחלקה ותכונות לפי התיאור הנ"ל.

לייצוג אוסף אפשר להשתמש במחלקות Stack, Queue, Node.
אפשר להוסיף תכונות נוספות.
חובה לתעד את התכונות.

(4 נק') ב. כתבו פעולה פנימית במחלקה **Floor** המקבלת מספר רישוי של מכונית ובודקת אם יש בקומה מקום חניה עבורה. אם כן, המכונית "תשובץ" לחניה באחד האזורים בקומה והפעולה תחזיר את צבע האזור.
אם לא, מכונית לא תשובץ והפעולה תחזיר מחרוזת "no room".

המחלקה חניון **(Parking)** מייצגת חניון. למחלקה יש תכונה אחת שהיא מערך עצמים מסוג Floor.

```
class Parking
{
  private Floor[] floors;
}
```

בכניסה לחניון עומדות מכוניות בהמתנה בתור. מכונית נכנסת רק אם יש מקום פנוי עבורה בחניון. אם נמצא מקום פנוי, המכונית יוצאת מ"תור ההמתנה" ומשובצת לחניון. אם לא נמצא מקום עבורה, היא נשארת בהמתנה.

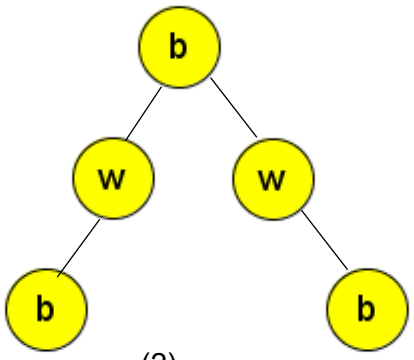
(5 נק') ג. כתבו פעולה המקבלת תור של מספרי רישוי של מכוניות שממתינות בכניסה. הפעולה תשבץ את המכוניות הממתינות בתור. מכוניות שלא נמצא עבורן מקום, תשארנה בתור ההמתנה.
עבור כל מכונית שנכנסה לחניון יודפסו מספר קומה וצבע אזור. הפעולה תסיים את עבודתה כאשר לא תשארנה מכוניות ב"תור המתנה" או כאשר לא יישאר מקום פנוי בחניון. בסיום העבודה הפעולה תדפיס הודעה: "ALL" – אם כל המכוניות שובצו במקומות חנייה, או "NOT ALL" – אם לא כולן שובצו למקום חנייה.

שאלה 11

עץ בינארי bt שכל צומת שבו הוא מטיפוס תו, ייקרא **כחול-לבן** אם הוא עץ ריק, או אם הוא מקיים את שלושת התנאים הבאים:

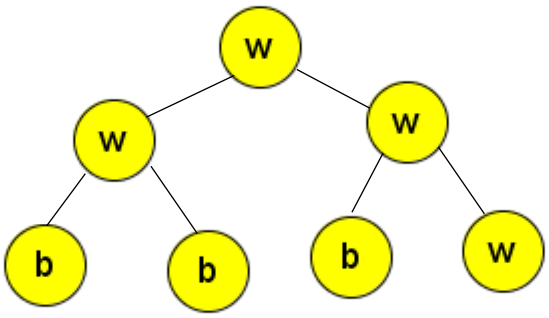
1. כל צומת הוא בעל ערך כחול- 'b' או לבן- 'w'.
2. כל עלה הוא כחול ('b').
3. אם ערך צומת שאינו עלה הוא כחול ('b'), אזי יש לו שני בנים לבנים ('w').

(4 נק') א. להלן ארבעה עצים בינאריים. לגבי כל אחד מהעצים, קבעו אם הוא עץ **כחול-לבן** או לא.
אם העץ אינו עץ **כחול-לבן**, העתיקו אותו למחברת, סמנו X בצמתים שאינם מקיימים את התנאים ונמקו מדוע אינם מקיימים.

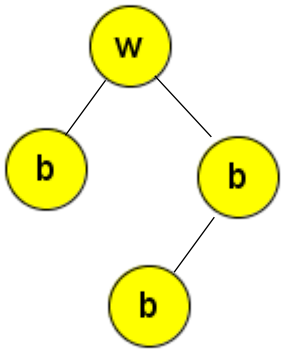


(2)

(1)



(4)



(3)

(6 נק') ב. כתבו פעולה שתקבל עץ בינארי **bt** ותחזיר **true** אם הוא עץ **כחול-לבן**, ולא הפעולה תחזיר **false**.

(2 נק') ג. מהי סיבוכיות זמן הריצה של הפעולה שכתבתם בסעיף ב'? נמקו את תשובתכם.

מבחן ב- C#

חלק א'

ענו על <u>שלוש</u> מבין השאלות 1-4 (ערך כל שאלה – 15 נקודות).

שאלה 1

מחסנית "שוות סכומים", היא מחסנית של מספרים שלמים המכילה מספר אי זוגי של איברים.

כך שכל שני איברים קיצוניים (ראשון + אחרון, שני + לפני אחרון וכו') שווים בסכומם לאיבר האמצעי מחסנית.

לדוגמה:

המחסנית הבאה היא "מחסנית שווה סכומים"

18	3	15	13	4	**25**	21	12	10	22	7

האיבר האמצעי במחסנית הוא 25. וכל זוג איברים קיצונים שווה ל- 25.

25=18+7 (האיבר הראשון והאיבר האחרון).

25=3+22 (האיבר השני והאיבר שלפני אחרון).

25=15+10

25=13+12

25=4+21

(12 נק') א. כתבו פעולה בשם `EqualSums` המקבלת מחסנית של מספרים שלמים. הפעולה מחזירה true אם המחסנית היא מחסנית "שווה סכומים", ואם היא אינה "שוות סכומים", הפעולה תחזיר false.

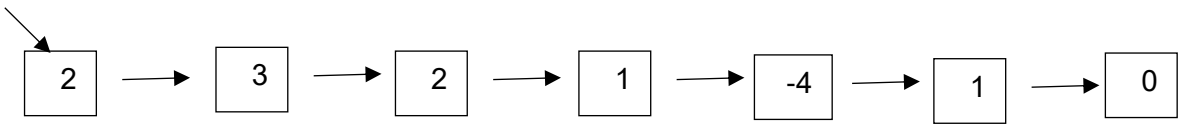
(3 נק') ב. מהי הסיבוכיות של הפעולה שכתבתם בסעיף א'? הסבירו את תשובתכם.

שאלה 2

נגדיר: "**רשימת מקטעים**" כשרשרת חוליות של מספרים שלמים ריקה או כשרשרת חוליות אשר כל אחת מהחוליות שלה מכילה ערך **קטן או שווה** למספר החוליות העוקבות לה (נמצאות אחריה בשרשרת).

לדוגמה:

השרשרת הבאה היא "רשימת מקטעים":



הערך של האיבר הראשון בשרשרת **(2)** הוא קטן מ**מספר** האיברים שאחריו (6).

הערך של האיבר החמישי בשרשרת **(-4)** הוא קטן מ**מספר** האיברים שאחריו (2).

(5 נק') א. כתבו פעולה חיצונית בשם NumNodesFollowing המקבלת הפנייה לחוליה בשרשרת כלשהי ומחזירה את מספר החוליות העוקבות לה.

(10 נק') ב. כתבו פעולה חיצונית בשם IsSection המקבלת הפניה לשרשרת חוליות מטיפוס שלם. הפעולה מחזירה true אם השרשרת היא "רשימת מקטעים", ולא, הפעולה תחזיר false.

שאלה 3

נתונות ארבע המחלקות הבאות:

```
public  class A
{
    protected int x;
    public A()
    {
        this.x = 1;
    }
    public A(int x)
    {
        this.x = x;
    }
    public virtual void Change()
    {
        this.x = this.x * 2;
    }
    public override string ToString()
    {
        return "X=" + this.x;
    }
}

public  class B : A
{
    protected int y;
    public B(int y)
    {
        this.y = -y;
    }
    public B(int x,int y): base(x)
    {
        this.y = -y;
    }
    public override string ToString()
    {
        return base.ToString() + " Y=" + this.y;
    }
}
```

```
public  class C : B
{
    private int z;
    public C(int x,int y,int z): base(x, y)
    {
        this.z = z;
    }
    public override void Change()
    {
        this.x = 3 * this.x;
        this.y = 3 * this.y;
        this.z = 3 * this.z;
    }
    public override string ToString()
    {
        return base.ToString() + " Z=" + this.z;
    }
}

public class TestABC
{
    public   static void Main(string[] args)
    {
        A[] data = new A[5];
        data [0] = new A (5);
        data [1] = new B(5,10);
        data [2] = new C(5,10,15);
        data [3] = data [0];
        data [4] = new B(5);

        for (int i = 0; i < data.Length; i++)
            Console.WriteLine(data[i]);

        for(int i=0;i< data.Length;i++)
            data [i].Change();
        for (int i = 0; i < data.Length; i++)
            Console.WriteLine(data[i]);
    }
}
```

עקבו אחרי ביצוע פעולה הראשית (Main) של המחלקה TestABC ורשמו מה יהיה הפלט של הפעולה הראשית.

חובה להציג עבור כל עצם שנוצר בפעולה את ערכי התכונות שלו.

שאלה 4

נתונות ארבע המחלקות הבאות:

```
public class B
{
    protected int num;
    public B( int num)
    {
        this.num = num;
    }
} // end of B

public class C : B
{
    public C(int num):base(num)
    {

    }
    public override bool Equals(Object other)
    {
        return ((other!=null) && (other is C) &&
        (num == ((C) other).num));
    }
 } // end of C

public class D : B
{
    public D( int num): base(num+1)
    {

    }

    public bool Equals(D other)
    {
        return ((other!=null) && (num == other.num));
    }
} // end of D
```

(4 נק׳) א. האם קיימת דריסה (overriding) או העמסה (overloading) של הפעולות?

הסבירו את תשובתכם.

נתונה הפעולה ראשית main במחלקה Program

```
public class Program
{
    public static void Main(string[] args)
    {
        B b = new B(1);
        C c = new C(1);
        D d = new D(1);
        B b1 = new D(1);
        Object c1 = new C(1);
        Object d1 = new D(1);
        (******)
     }
}
```

(3 נק׳) ב. עקבו אחרי ביצועה וציינו מהם העצמים שנוצרו. עבור כל עצם יש להציג את ערך תכונת ה- num שלו.

(8 נק׳) ג. בכל אחד מסעיפים הבאים השורה (*****) תוחלף בשורת קוד משורות הקוד 1-4.

כתבו בעבור כל אחד מהסעיפים מה יהיה הפלט ואיזו מהפעולות Equals תופעל.

שימו לב שאין קשר בין סעיפים!

```
1)  Console.WriteLine(b1.Equals(b));
2)  Console.WriteLine (c1.Equals(c));
3)  Console.WriteLine (d1.Equals(d));
4)  Console.WriteLine (d.Equals((D)d1));
```

חלק ב'

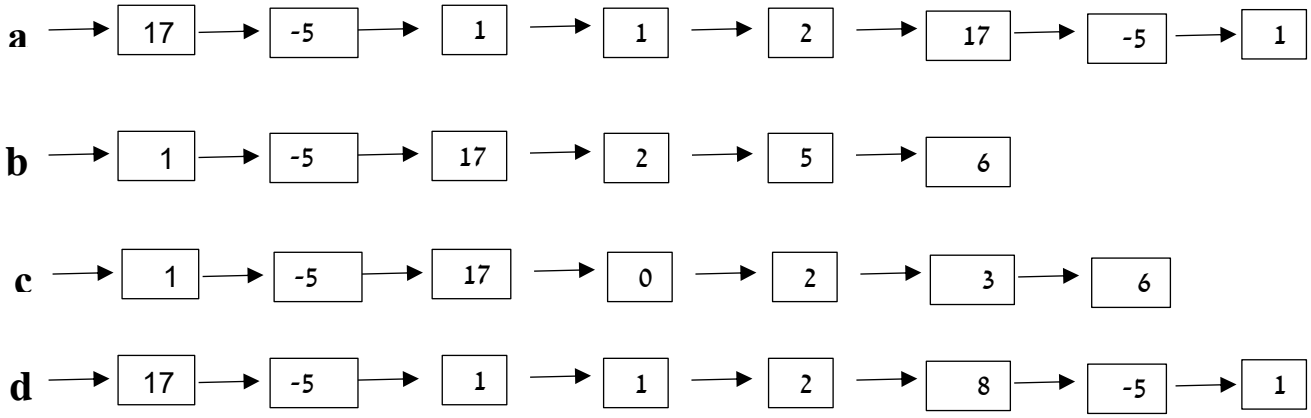
ענו על <u>שתיים</u> מבין השאלות 5-8 (ערך כל שאלה – 15 נקודות).

שאלה 5

נתונה הפעולה הבאה:

```
public static void what(Node<int>f, Node<int> s)
{
  Stack<int>st = new Stack<int>();
  while(s!=null)
  {
     st.Push(s.GetValue());
     s = s.GetNext();
  }
  while(f!=null && !st.IsEmpty() && (st.Pop()==f.GetValue()))
  {
     Console.Write(f.GetNext()+", ");
     f = f.GetNext();
  }
    Console.WriteLine("STOP!");
}
```

נתונות ארבע השרשרות הבאות:



(6 נק') א. עקבו אחרי זימון הפעולה `What(a, b)` ורשמו מה יהיה הפלט של הזימון. <u>יש להציג את המעקב</u>!

(6 נק') ב. עקב זימון הפעולה What התקבל הפלט (משמאל לימין):

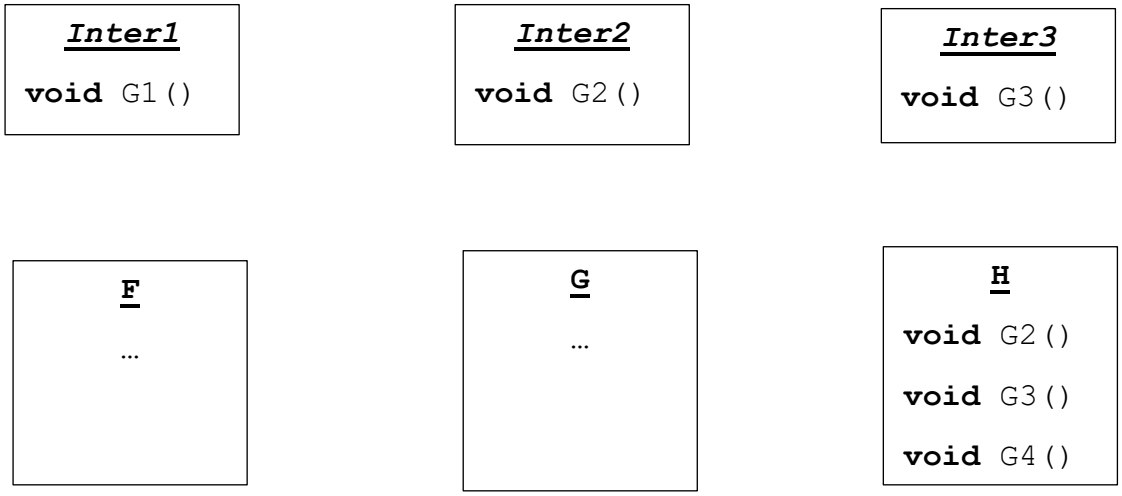
```
1, -5, 17, STOP!
```

אילו פרמטרים נשלחו לפעולה? יש לבחור זוג פרמטרים מתוך ארבע השרשרות הנתונות **(a, b, c, d)**.

(3 נק') ג. מה מבצעת הפעולה באופן כללי?

שאלה 6

לפניכם שלושה ממשקים ושלוש מחלקות:



נתון קטע מפעולה ראשית. הקטע עובד <u>ללא שגיאות הידור או שגיאות זמן ריצה</u>.

```
Inter1 x = new F();
G y = new G();
Inter3 z = new H();
x = y;
y = new H();
y.G2();
z.G3();
((H)z).G4();
z = new F();
```

(8 נק') א. השלימו את תרשים UML: הוספו כותרות הפעולות במחלקות F ו- G, ציירו קשרים בין המחלקות והממשקים.

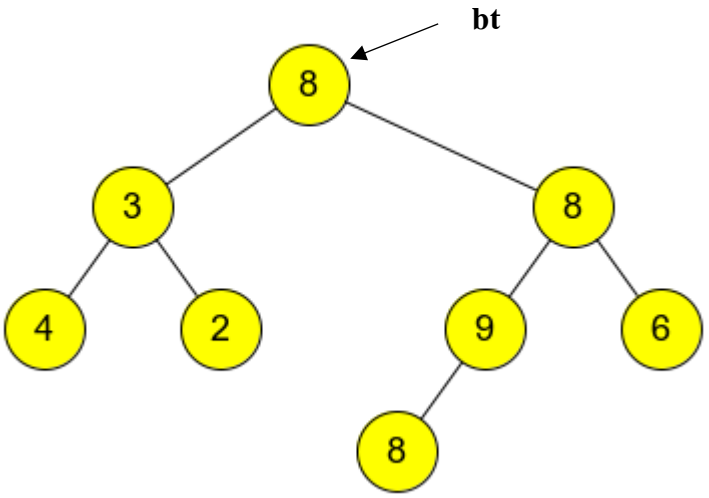
(7 נק') ב. כתבו פעולה המקבלת מערך עצמים מסוג Object. הפעולה מדפיסה את מספר האיברים שעבורם אפשר להפעיל את הפעולה **G1()** ואת מספר האיברים שעבורם <u>אי אפשר</u> להפעיל את הפעולה **G3()**.

שאלה 7

לפניכם ארבע פעולות:

```
public static int First(BinNode<int> bt){
    if(bt==null)
        return 0;
    return
       bt.GetValue()+First(bt.GetLeft())+First(bt.GetRight());
}
public static int Second(BinNode<int> bt){
    return First(bt.GetLeft())+First(bt.GetRight());
}
public static void Third(BinNode<int> bt){
    if(bt!=null)
    {
      bt.SetValue(Second(bt));
      Third(bt.GetLeft());
      Third(bt.GetRight());
    }
}
```

נתון העץ הבינרי הבא:



(5 נק') א. מה תהיה תוצאת זימון הפעולה **Second(bt)**? **חובה להראות את המעקב!**

(3 נק') ב. מה מחזירה הפעולה **Second(BinNode<int> bt)** באופן כללי?

(5 נק') ג. מה תהיה תוצאת הזימון **Third(bt)** לעץ הנתון **bt**?

חובה להראות את המעקב אחרי הפעולה **Third**. אין צורך במערב אחרי הפעולה Second.

(2 נק') ד. מה מבצעת הפעולה **Third(BinNode<int> bt)** באופן כללי?

שאלה 8

נתונה המחלקה **MyNode** הבאה:

```
public class MyNode
{
    private int value;
    private int howManyBig;
    private MyNode next;
```

למחלקה הוגדר בנאי:

```
   public MyNode(int val)
    {
      this.value = val;
      this.howManyBig = 0;
      this.next = null;
    }
```

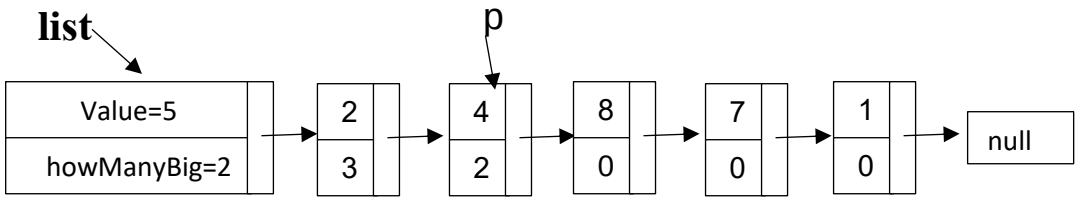
לכל תכונה במחלקה הוגדרו פעולות Get/Set

```
}
```

בעזרת המחלקה MyNode נבנית שרשרת חוליות לפי הכלל הבא:

כל חוליה היא מסוג MyNode. בכל איבר התכונה value מאחסנת מספר שלם, התכונה howManyBig מאחסנת מספר שלם ששווה ל**מספר** האיברים בשרשרת שנמצאים **אחרי** האיבר הנוכחי, שערך התכונה value שלהם גדול מערך value של האיבר הנוכחי. התכונה next מחזיקה מצביע על האיבר הבא.

לדוגמה:



P מצביע על האיבר שהתכונה value שלו שווה ל- 4, והתכונה howManyBig שווה ל- 2 כי יש שני איברים אחריו, שערכי ה- value שלהם גדולים מ- 4 (כלומר 8 ו- 7).

(12 נק') א. כתבו פעולה חיצונית המקבלת הפניה לחוליה הראשונה של שרשרת החוליות מסוג MyNode, מספר שלם val ומספר שלם וחיובי position. כותרת הפעולה:

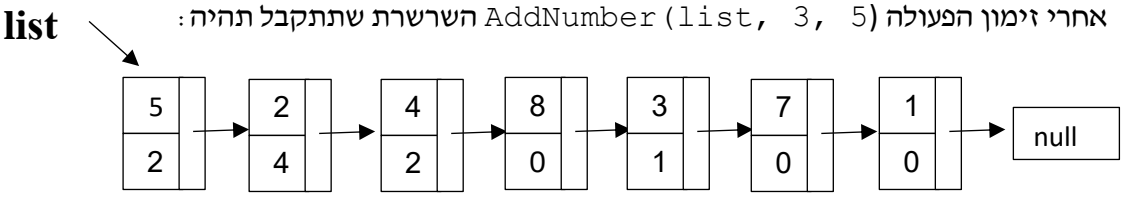
```
public static MyNode AddNumber(MyNode list, int val, int position)
```

הפעולה צריכה להכניס איבר חדש לשרשרת במיקום position (המיקום של האיבר הראשון בשרשרת הוא 1). ערך התכונה value של האיבר החדש יהיה שווה לפרמטר val.

השרשרת שתתקבל צריכה לשמור על החוקים של השרשרת המקורית לגבי התכונה howManyBig.

לדוגמה:

אחרי זימון הפעולה AddNumber(list, 3, 5) השרשרת שתתקבל תהיה:



(3 נק') ב. מהי סיבוכיות הפעולה? הסבירו את תשובתכם.

חלק ג'

ענו על <u>שתיים</u> מבין השאלות 9-11 (ערך כל שאלה – 12 נקודות).

שאלה 9

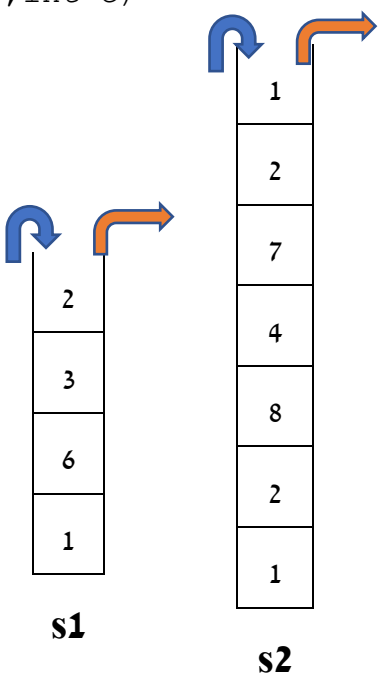
נתונות שתי הפעולות הרקורסיביות **Secret** ו- **Mystery**:

הפעולה **Secret** מקבלת מחסנית לא ריקה של מספרים שלמים.

```
public static int Secret (Stack<int> s)
{
   int x=s.Pop();
   if(s.IsEmpty())
     return x;
   int y = Secret(s);
   s.Push(x);
   return y;
}
```

הפעולה Mystery מקבלת מחסנית של מספרים שלמים ומספר שלם וחיובי c.

```
 public static bool Mystery (Stack<int>s,int c)
 {
     if (c==0 ||  s.IsEmpty())
          return true;
     int y=Secret(s);
     if (s.IsEmpty())
          return false;
     int x=s.Pop();
     if (x==y)
          return Mystery (s, c-1);
     return false;
}
```



כמו כן, נתונות המחסניות s1 ו-s2.

(3 נק') א. עקבו אחר זימון הפעולה Secret עבור המחסנית s1. מהו הערך שיוחזר מהפעולה?

יש להראות בכל שלב את הערכים של x, y ואת המחסנית s1.

(2 נק') ב. מהי מטרת הפעולה **Secret**?

(5 נק') ג. עקבו אחר זימון הפעולה **Mystery(s2, 2)** ורשמו מהו הערך שיוחזר מהפעולה.

יש להראות בכל שלב את הערכים של y, x, c, ואת המחסנית s. אין צורך לעקוב אחרי **Secret**.

(2 נק') ד. מהי מטרת הפעולה **Mystery**?

שאלה 10

<u>הערה</u>: הניחו שהפעולות Get/Set הוגדרו בעבור כל תכונה בכל אחת מהמחלקות.
אם הוספתם פעולות עזר נוספות, יש לציין את המחלקה שבה נמצאת פעולת העזר.

בחניון "לב העיר" קיימות מספר קומות. בכל קומה יש אזורים המזוהים לפי צבע. מספר האזורים אינו שווה בהכרח בכל קומה.

כל **אזור (Area)** מאופיין באמצעות:

- צבע.
- מספר מקומות פנויים כרגע.
- רשימת מספרי הרישוי של המכוניות שחונות בו כרגע (סדר המכוניות לא משנה).

כל **קומה (Floor)** מאופיינת על ידי מספר הקומה ואוסף האזורים שבה.

(3 נק') א. כתבו מחלקות עבור הטיפוסים: **אזור** – **Area** ו**קומה** – **Floor**.
לכל מחלקה כתבו כותרת מחלקה ותכונות לפי התיאור הנ"ל.

לייצוג אוסף אפשר להשתמש במחלקות Stack, Queue, Node.
אפשר להוסיף תכונות נוספות.
חובה לתעד את התכונות.

(4 נק') ב. כתבו פעולה פנימית במחלקה **Floor** המקבלת מספר רישוי של מכונית ובודקת אם יש בקומה מקום חניה עבורה. אם כן, המכונית "תשובץ" לחניה באחד האזורים בקומה והפעולה תחזיר את צבע האזור.
אם לא, מכונית לא תשובץ והפעולה תחזיר מחרוזת "no room".

המחלקה חניון **(Parking)** מייצגת חניון. למחלקה יש תכונה אחת שהיא מערך עצמים מסוג Floor.

```
class Parking
{
  private Floor[] floors;
}
```

בכניסה לחניון עומדות מכוניות בהמתנה בתור. מכונית נכנסת רק אם יש מקום פנוי עבורה בחניון. אם נמצא מקום פנוי, המכונית יוצאת מ"תור ההמתנה" ומשובצת לחניון. אם לא נמצא מקום עבורה, היא נשארת בהמתנה.

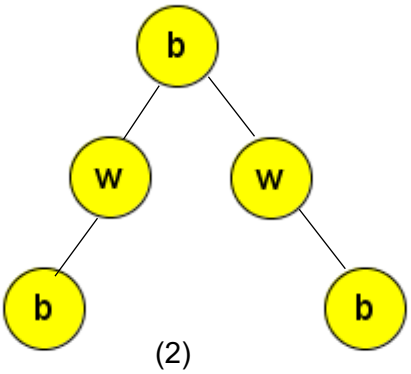
(5 נק') ג. כתבו פעולה המקבלת תור של מספרי רישוי של מכוניות שממתינות בכניסה. הפעולה תשבץ את המכוניות הממתינות בתור. מכוניות שלא נמצא עבורן מקום, תשארנה בתור ההמתנה.
עבור כל מכונית שנכנסה לחניון יודפסו מספר קומה וצבע אזור. הפעולה תסיים את עבודתה כאשר לא תשארנה מכוניות ב"תור המתנה" או כאשר לא יישאר מקום פנוי בחניון. בסיום העבודה הפעולה תדפיס הודעה: "ALL" – אם כל המכוניות שובצו במקומות חנייה, או "NOT ALL" – אם לא כולן שובצו למקום חנייה.

שאלה 11

עץ בינארי bt שכל צומת שבו הוא מטיפוס תו, ייקרא **כחול-לבן** אם הוא עץ ריק, או אם הוא מקיים את שלושת התנאים הבאים:

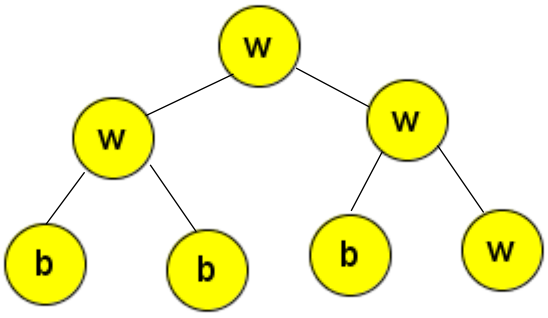
1. כל צומת הוא בעל ערך כחול- 'b' או לבן- 'w'.
2. כל עלה הוא כחול ('b').
3. אם ערך צומת שאינו עלה הוא כחול ('b'), אזי יש לו שני בנים לבנים ('w').

(4 נק') א. להלן ארבעה עצים בינאריים. לגבי כל אחד מהעצים, קבעו אם הוא עץ **כחול-לבן** או לא.
אם העץ אינו עץ **כחול-לבן**, העתיקו אותו למחברת, סמנו X בצמתים שאינם מקיימים את התנאים ונמקו מדוע אינם מקיימים.

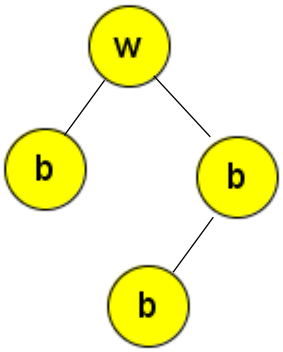


(2)

(1)



(4)



(3)

(6 נק') ב. כתבו פעולה שתקבל עץ בינארי **bt** ותחזיר **true** אם הוא עץ **כחול-לבן**, ולא הפעולה תחזיר **false**.

(2 נק') ג. מהי סיבוכיות זמן הריצה של הפעולה שכתבתם בסעיף ב'? נמקו את תשובתכם.

בהצלחה!

נספח לשאלון 97105 – מבני נתונים ותכנות מונחה עצמים – JAVA

נספח ממשקים מבנה הנתונים בתוכנית הלימודים

ממשק המחלקה חוליה הגנרית- Node<T>

המחלקה מגדירה חוליה גנרית שבה ערך מטיפוס T והפניה לחוליה העוקבת.

הפעולה בונה חוליה. הערך של החוליה הוא x, ואין לה חוליה עוקבת.	`Node (T x)`
הפעולה בונה חוליה. הערך של החוליה הוא x, והחוליה העוקבת לה היא next. ערכו של next יכול להיות **null**.	`Node (T x, Node<T> next)`
הפעולה מחזירה את הערך של החוליה.	`T getValue()`
הפעולה מחזירה את החוליה העוקבת. אם אין חוליה עוקבת, הפעולה מחזירה **null**.	`Node<T> getNext()`
הפעולה משנה את הערך השמור בחוליה ל- x.	`void setValue (T x)`
הפעולה מחזירה **true** אם יש חוליה נוספת.	`boolean hasNext()`
הפעולה משנה את החוליה העוקבת ל- next. ערכו של next יכול להיות **null**.	`void setNext (Node<T> next)`
הפעולה מחזירה מחרוזת המתארת את החוליה.	`String toString()`

יעילות הפעולות: כל הפעולות מתבצעות בסדר גודל קבוע, O(1).

ממשק המחלקה הגנרית - מחסנית Stack<T>

המחלקה מגדירה טיפוס אוסף בעל פרוטוקול **LIFO** להכנסה והוצאה של ערכים.

הפעולה בונה מחסנית ריקה.	**Stack**()
הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אם היא אינה ריקה.	**boolean** isEmpty()
הפעולה מכניסה את הערך x לראש המחסנית הנוכחית (דחיפה).	**void** push (T x)
הפעולה מוציאה את הערך שבראש המחסנית הנוכחית ומחזירה אותו (שליפה). **הנחה:** המחסנית הנוכחית אינה ריקה.	**T** pop()
הפעולה מחזירה את הערך שבראש המחסנית הנוכחית מבלי להוציאו. **הנחה**: המחסנית הנוכחית אינה ריקה.	**T** top()
הפעולה מחזירה תיאור של המחסנית, כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש המחסנית): $[x_1, x_2, \ldots, x_n]$	**String** toString()

יעילות הפעולות- מחלקה מיוצגת בעזרת שרשרת חוליות.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה toString() המתבצעת בסדר גודל לינארי.

ממשק המחלקה הגנרית- תור Queue<T>

המחלקה מגדירה טיפוס אוסף עם פרוטוקול **FIFO** להכנסה והוצאה של ערכים.

הפעולה בונה תור ריק.	**Queue**()
הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ו"שקר" אם הוא אינו ריק.	**boolean** isEmpty()
הפעולה מכניסה את הערך x לסוף התור הנוכחי.	**void** insert (Tx)
הפעולה מוציאה את הערך שבראש התור הנוכחי ומחזירה אותו. **הנחה**: התור הנוכחי אינו ריק.	**T** remove()
הפעולה מחזירה את ערכו של האיבר שבראש התור מבלי להוציאו. **הנחה**: התור הנוכחי אינו ריק	**T** head()
הפעולה מחזירה מחרוזת המתארת את התור כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש התור): $[x_1, x_2, \ldots, x_n]$	**String** toString()

יעילות הפעולות- המחלקה מיוצגת בעזרת שרשרת חוליות והפניה לזנב התור.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה toString() המתבצעת בסדר גודל לינארי.

נספח לשאלון 97105 – מבני נתונים ותכנות ונחה עצמים – C#

נספח ממשקים מבנה הנתונים בתוכנית הלימודים

ממשק המחלקה חוליה הגנרית- Node<T>

המחלקה מגדירה חוליה גנרית שבה ערך מטיפוס T והפניה לחוליה העוקבת.

הפעולה בונה חוליה. הערך של החוליה הוא x, ואין לה חוליה עוקבת.	**Node** (T x)
הפעולה בונה חוליה. הערך של החוליה הוא x, והחוליה העוקבת לה היא next. ערכו של next יכול להיות **null**.	Node (T x, Node<T> next)
הפעולה מחזירה את הערך של החוליה.	**T** GetValue()
הפעולה מחזירה את החוליה העוקבת. אם אין חוליה עוקבת, הפעולה מחזירה **null**.	**Node<T>** GetNext()
הפעולה משנה את הערך השמור בחוליה ל- x.	**void** SetValue (T x)
הפעולה מחזירה **true** אם יש חוליה נוספת.	**bool** HasNext()
הפעולה משנה את החוליה העוקבת ל-next. ערכו של next יכול להיות **null**.	**void** SetNext (Node<T> next)
הפעולה מחזירה מחרוזת המתארת את החוליה.	**override string** ToString()

יעילות הפעולות: כל הפעולות מתבצעות בסדר גודל קבוע, O(1).

ממשק המחלקה הגנרית - מחסנית Stack<T>

המחלקה מגדירה טיפוס אוסף בעל פרוטוקול **LIFO** להכנסה והוצאה של ערכים.

הפעולה בונה מחסנית ריקה.	**Stack**()
הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אם היא אינה ריקה.	**bool** IsEmpty()
הפעולה מכניסה את הערך x לראש המחסנית הנוכחית (דחיפה).	**void** Push (T x)
הפעולה מוציאה את הערך שבראש המחסנית הנוכחית ומחזירה אותו (שליפה). **הנחה**: המחסנית הנוכחית אינה ריקה.	**T** Pop()
הפעולה מחזירה את הערך שבראש המחסנית הנוכחית בלי להוציאו. **הנחה**: המחסנית הנוכחית אינה ריקה.	**T** Top()
הפעולה מחזירה תיאור של המחסנית, כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש המחסנית): $[x_1, x_2, \ldots, x_n]$	**override string** ToString()

יעילות הפעולות- מחלקה מיוצגת בעזרת שרשרת חוליות.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה () ToString המתבצעת בסדר גודל לינארי.

ממשק המחלקה הגנרית - תור Queue<T>

המחלקה מגדירה טיפוס אוסף עם פרוטוקול **FIFO** להכנסה והוצאה של ערכים.

הפעולה בונה תור ריק.	**Queue**()
הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ו"שקר" אם הוא אינו ריק.	**bool** IsEmpty()
הפעולה מכניסה את הערך x לסוף התור הנוכחי.	**void** Insert (Tx)
הפעולה מוציאה את הערך שבראש התור הנוכחי ומחזירה אותו. **הנחה**: התור הנוכחי אינו ריק.	**T** Remove()
הפעולה מחזירה את ערכו של האיבר שבראש התור מבלי להוציאו. **הנחה**: התור הנוכחי אינו ריק.	**T** Head()
הפעולה מחזירה מחרוזת המתארת את התור כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש התור): $[x_1, x_2, \ldots, x_n]$	**override string** ToString()

יעילות הפעולות- המחלקה מיוצגת בעזרת שרשרת חוליות והפניה לזנב התור.

כל הפעולות מתבצעות בסדר גודל קבוע, O(1), למעט הפעולה () ToString המתבצעת בסדר גודל לינארי.

תרגום לערבית שאלון 97105 – מבנה נתונים ותכנות מונחה עצמים – מועד ב' קיץ 22

قاموس مُساعد – امتحان معهد العلوم والتكنولوجيا 97105 موعد ب سنة 2022

القسم أ

رقم السؤال	الكلمة \ التعبير بالعبرية	الكلمة \ التعبير بالعربية
1	אין צורך במילון	لا حاجة الى قاموس
2	עוקבות	المتتالية
3	אין צורך במילון	لا حاجة الى قاموس
4	אין צורך במילון	لا حاجة الى قاموس

القسم ب

رقم السؤال	الكلمة \ التعبير بالعبرية	الكلمة \ التعبير بالعربية
5	אין צורך במילון	لا حاجة الى قاموس
6	אין צורך במילון	لا حاجة الى قاموس
7	אין צורך במילון	لا حاجة الى قاموس
8	אין צורך במילון	لا حاجة الى قاموس

القسم ج

رقم السؤال	الكلمة \ التعبير بالعبرية	الكلمة \ التعبير بالعربية
9	אין צורך במילון	لا حاجة الى قاموس
10	חניון	موقف سيارات
10	חניה (ברבים-חניות)	وقوف سيارة (الجمع وقوف سيارات)
10	פנוי (ברבים-פנויים)	مُتفرغ (الجمع مُتفرغون)
10	קומה (ברבים-קומות)	طابق (الجمع طوابق)
10	צבע	لون
10	"תשובץ"	وضع
10	המתנה	انتظار
11	אין צורך במילון	لا حاجة الى قاموس

מבני נתונים ותכנות מונחה עצמיים 97105 – מועד ב' קיץ 2022

שאלה	סעיף	תת-סעיף	ניקוד	הערות
1	א	-	12	חריגה – להוריד 2 נק' לולאה אין סופית – להוריד 2 נק'
	ב	-	3	בלי הסבר – לא לתת נק'
2	-	-	5	
			10	אם לא השתמש בסעיף א' – אין להוריד נקודות
3		-	15	בלי מעקב לא לתת נקודות יש להראות שיני בערכי תכונות אחרי הפעולה CHANGE
4	א	-	4	יש להתייחס לדריסה/העמסה של EQUALS ממחלקה OBJECT
	ב	-	3	כל סעיף – 0.5 נק'
	ג		8	כל סעיף – 2 נק'
5	א	-	6	בלי מעקב לא לתת נקודות
	ב		6	יש להתאים תשובות בסעיפים ב' וג'
	ג	-	3	
6	א	-	8	• עבור כל חץ חסר/מיותר להוריד 1 נק' • אם שגה בכיין חץ הירושה/מימוש – להוריד 3 נק' פעם אחת
	ב	-	7	

שאלה	סעיף	תת-סעיף	ניקוד	הערות
7	א	-	5	בלי מעקב לא לתת נקודות
	ב	-	3	
	ג		5	בלי מעקב לא לתת נקודות
	ד		2	
8	א	-	12	הוספה 6 עדכון תכונה חוליות לפני חדשה – 3 נק' עדכון תכונה לחוליה חדשה תאחרי הכנסה – 3 נק'
	ב	-	3	בלי הסבר לא לתת נקודות
9	א	-	3	בלי מעקב לא לתת נקודות
	ב		2	
	ג	-	5	בלי מעקב לא לתת נקודות
	ד		2	
10	א	-	3	ניתן להשתמש בכל סוג של מבנה נתונים!
	ב	-	4	
	ג		5	
11	א		4	כל סעיף 1 נקודות
	ב		6	
	ג		2	בלי הסבר לא לתת נקודות