משרד העבודה
המכון הממשלתי להכשרה טכנולוגית

מה"ט
המכון הממשלתי להכשרה טכנולוגית

מועד הבחינה: קיץ תשפ"ג – 2023 – מועד א'
מספר השאלון: 97105
נספח ממשקים לבחינה JAVA
נספח ממשקים לבחינה C#
מילון עזר

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 16)

בכל מבחן 10 שאלות.

חלק א' – 48 נקודות

שאלות 1-4: יש לענות על **שלוש** שאלות בלבד. ערך כל שאלה 16 נקודות.

חלק ב' – 36 נקודות

שאלות 5-8: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 18 נקודות.

חלק ג' – 16 נקודות

שאלות 9-10: יש לענות על **אחת** בלבד. ערך שאלה 16 נקודות.

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 29 עמודי בחינה ו- 7 עמודי נספחים.

מבחן ב- JAVA

חלק א'

ענו על <u>שלוש</u> מבין השאלות 1-4 (ערך כל שאלה – 16 נקודות).

שאלה 1

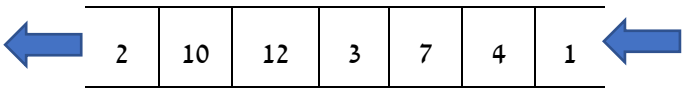
(6 נק') א. כתבו פעולה `putInPlace` המקבלת תור `q` ומספר `num` וממקמת את המספר `num` במיקומו, כך שכל האיברים הקטנים ממנו, נמצאים לפניו וכל האיברים השווים לו או הגדולים ממנו נמצאים אחריו.

(אין משמעות לסדר של האיברים הקטנים או הגדולים ממנו).

הפעלה <u>תחזיר</u> את מיקומו הסידורי של המספר `num` בתוך התור <u>אחרי הכנסתו</u>.

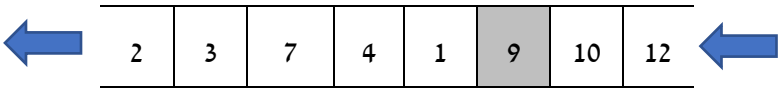
לדוגמא:

עבור התור q הבא:



והמספר `num=9`

התור שיתקבל <u>יכול להיות</u>:



והפעולה תחזיר **6** (מיקומו של ה-**num** אחרי הכנסתו).

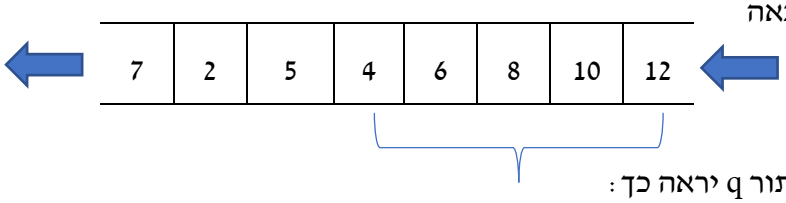
כותרת הפעולה:

```
public static int putInPlace(Queue<Integer> q, int num)
```

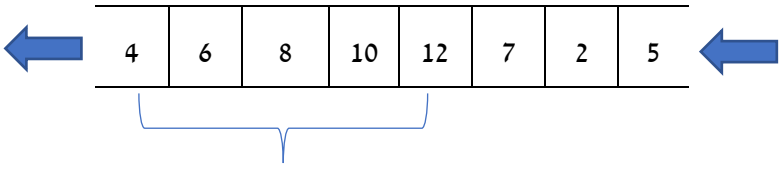
(6 נק') ב. כתבו פעולה חיצונית המקבלת תור `q` של מספרים שלמים ומספר שלם וחיובי `k`, ומעבירה את **האיברים האחרונים בתור אל ראש התור**. אפשר להניח ש-`k` קטן מאורך התור.

לדוגמה:

עבור התור `q` הבאה



והמספר `k=5` התור `q` יראה כך:



כותרת הפעולה:

```
public static void moveToFront(Queue<Integer> q, int k)
```

(4 נק') ג. מהי הסיבוכיות של הפעולות שכתבתם בסעיפים א' ו-ב'? **הסבירו את תשובתכם.**

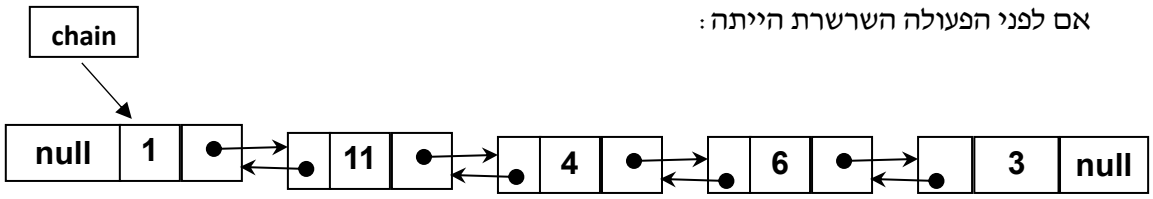
שאלה 2

בעזרת המחלקה `BinNode` אפשר לממש שרשרת חוליות דו-כיוונית.

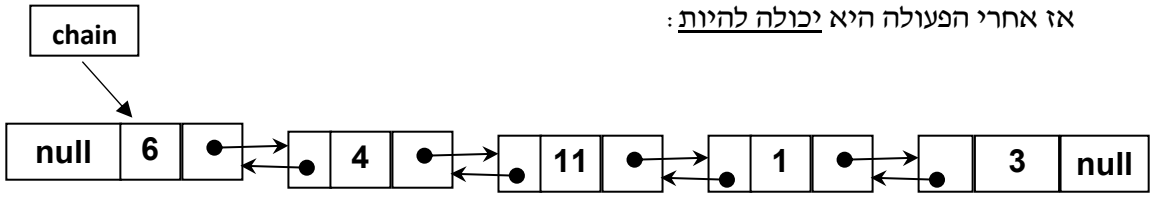
(12 נק׳) א. כתבו פעולה המקבלת הפנייה לחוליה הראשונה (הכי שמאלית) של שרשרת חוליות הדו-כיוונית ומסדרת אותה כך שכל הערכים הזוגיים יהיו בתחילת השרשרת וכל הערכים האי-זוגיים יהיו בסופה. סדר האיברים לא חשוב.

לדוגמה:

אם לפני הפעולה השרשרת הייתה:



אז אחרי הפעולה היא יכולה להיות:



שימו לב: אין להשתמש במבנה נתונים נוסף!

כותרת הפעולה:

```
public static void order(BinNode<Integer> chain)
```

(4 נק׳) ב. מהי סיבוכיות הפעולה `order` שכתבתם בסעיף א׳? **הסבירו את תשובתכם.**

שאלה 3

נתונות שתי המחלקות Flower ו- Rose

```
public class Flower
{
  protected  int  height;
  private int price;
  public  Flower (int  val) { this.height = val; this.price = 10;}
  public  int  getHeight()  { return this.height; }
  public  int  getPrice ()  { return this.price; }
}
public class Rose extends Flower
{
  private  String  color;
  public Rose(int val, String col)
    {
        super(val);
        this.color=col;
    }
  public  boolean  validHeight()
    {
       return this.height > 10 && this.height < 30;
    }
}
```

(5 נק') א. לפניכם חמישה היגדים. קבעו לכל אחד מהם אם הוא נכון או אינו נכון, ונמקו את קביעתכם.

1. המחלקה **Flower** יורשת את הפעולה `validHeight()` מהמחלקה **Rose**.
2. המחלקה **Rose** יורשת את כל התכונות ואת כל הפעולות של המחלקה **Flower**.
3. המחלקה **Rose** יכולה לגשת ישירות לתכונה `height` של המחלקה **Flower**.
4. המחלקה **Flower** יכולה לגשת לתכונה `color` של המחלקה **Rose**.
5. לעצמים מטיפוס **Rose** אין תכונה `price`.

(5 נק') ב. לפניכם קטע קוד מהתוכנית הראשית:

```
Flower  first = new Rose (20, "RED");
Flower  second = new Flower (93);
```

בעבור כל אחת מההוראות שלפניך קבעו אם היא תקינה או אינה תקינה.

אם היא אינה תקינה, נמקו את קביעתכם וכתבו אם זו שגיאת ריצה או שגיאת הידור (קומפילציה).

```
1. boolean b = first. validHeight();
2. boolean b = second. validHeight();
3. boolean b = ((Rose)first). validHeight();
4. boolean b = ((Rose)second). validHeight();
5. boolean b = first.getPrice() == second.price;
```

(6 נק') ג. כתבו פעולה המקבלת מערך עצמים מטיפוס Object. על הפעולה להדפיס כמה עצמים הם מטיפוס **Rose**, כמה עצמים מטיפוס **Flower** ואינו מטיפוס **Rose** וכמה עצמים הם לא מטיפוס **Flower**.

שאלה 4

בפרויקט מסוים הוגדרו שלשה ממשקים ושתי מחלקות:

```
public interface Dancer{
    void lonelyDance();
    void coupleDance(Dancer d);
}
public interface Singer  {    void sing(); }
public interface Musician {   void play(); }
public class Actor implements Dancer, Singer, Musician
{
    private String name;
    private int age;
    …
}
public class Driver
{
    public static void main(String[] args)
    {
        Actor actor1 = new Actor("Bob", 25);
        Actor actor2 = new Actor("Alice");
        ( *****)
    }
}
```

(4 נק׳) א. השלימו את המחלקה Actor: תכתבו כותרות של הפעולות החסרות בה.

(6 נק׳) ב. בכל אחד מסעיפים הבאים השורה (*****) תוחלף בשורת קוד משורות הקוד 1-6. כתבו, עבור כל אחד מהסעיפים, אם הקוד תקין או לא. אם הקוד אינו תקין – יש להסביר למה הוא אינו תקין ולציין את סוג השגיאה (קומפילציה או זמן ריצה).

שימו לב שאין קשר בין סעיפים!

```
(1)   Dancer dancer1 = actor1;
      dancer1. coupleDance (actor2);

(2)   Singer singer2 = new Singer();
      singer2.sing();

(3)   Singer singer3 = new Actor ("Clark",30);
      Dancer dancer3 = (Actor)singer3;

(4)   Singer actor4 = new Actor("Danny");
      actor4.play();

(5)   Dancer actor5 = new Actor("Eddie");
      Singer singer5 = (Singer) actor5 ;
      singer5.sing();

(6)   Musician musician6 = new Actor("Freddie", 45);
      ((Actor) musician6).lonelyDance();
      ((Singer) musician6).sing();
```

(6 נק') ג. **(אין קשר לסעיפים א' ו-ב')**

נתונה הפעולה main הבאה:

```
public static void main(String[] args)
{
   C c = new A();
   B b1 = (B)(new A());
   B b2 = new D();
   A a = new D();
}
```

עבור כל אחת מן האפשרויות 1-6, קבעו אם יחס ההורשה בין המחלקות מתאים לקוד של הפעולה main הנתונה.

הסבירו את תשובתכם.

1. C יורשת מ- A, B יורשת מ- A, D יורשת מ- A.
2. A יורשת מ- C, B יורשת מ- A, D יורשת מ- A.
3. C יורשת מ- A, B יורשת מ- A, D יורשת מ- B.
4. A יורשת מ- C, C יורשת מ- B, D יורשת מ- A.
5. A יורשת מ- C, B יורשת מ- A, D יורשת מ- B.
6. A יורשת מ- C, B יורשת מ- C, D יורשת מ- C.

חלק ב׳

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

שאלה 5

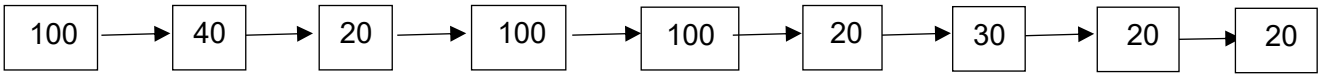
נתונה רשימה של מספרים שלמים. כל איבר (מספר) ברשימה יכול להופיע פעם אחת או יותר.

רשימת תדירויות (Frequency List) היא רשימה של מספרים שלמים שבה כל מספר מופיע <u>פעם אחת</u> 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.**

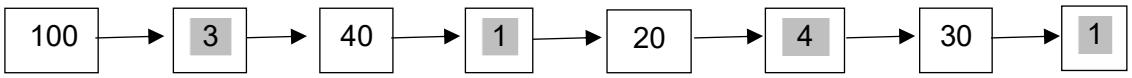
שימו לב: הפעולה <u>תעדכן</u> את הרשימה שהתקבלה כך שכל ערך יופיע פעם אחת ולאחריו מספר הפעמיים שהופיע ברשימה המקורית.
יש למחוק ערכים שחוזרים על עצמם חוץ מהמופע הראשון.
אין להשתמש במבני נתונים נוספים!!!

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 <u>המופיע הכי הרבה</u> 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.

שאלה 6

במלון ״בא ונח״ אפשר להזמין שלושה סוגים שונים של חדרים – חדר רגיל, חדר משפחתי וסוויטה.
מערכת ניהול המלון כוללת ארבע מחלקות: `Room`, `FamilyRoom`, `Suite`, `Hotel`.

לחדר רגיל (`Room`) יש את התכונות הבאות:

- `num` – מספר חדר, מטיפוס שלם, `int`
- `beds` – מספר המיטות, מטיפוס שלם, `int`
- `price` – מחיר, מטיפוס ממשי, `double`

לחדר משפחה (`FamilyRoom`) יש את התכונות הבאות:

- `num` – מספר חדר, מטיפוס שלם, `int`
- `beds` – מספר המיטות, מטיפוס שלם, `int`
- `price` – מחיר, מטיפוס ממשי, `double`
- `babyBed` – האם נוספה מיטת תינוק, מטיפוס `boolean` (`true` – כן נוספה מיטה, `false` – לא)
- `balcony` – האם יש מרפסת, מטיפוס `boolean` (`true` – יש, `false` – אין)

לסוויטה (`Suite`) יש את התכונות הבאות:

- `num` – מספר חדר, מטיפוס שלם, `int`
- `beds` – מספר המיטות, מטיפוס שלם, `int`
- `price` – מחיר, מטיפוס ממשי, `double`
- `jacuzzi` – האם יש ג׳קוזי, מטיפוס `boolean` (`true` – יש, `false` – אין)

(2 נק׳) א. סרטטו תרשים UML המתאר את הקשר בין המחלקות: `Room`, `FamilyRoom`, `Suite`
באופן <u>המתאים ביותר לעקרונות של תכנות מונחה עצמים</u>.

(3 נק׳) ב. לכל אחת מהמחלקות `Room`, `FamilyRoom`, `Suite` כתבו כותרת המחלקה ותכונות שלה.
אפשר להניח שבכל מחלקה הוגדרו פעולות `set/get`

(6 נק׳) ג. לאחר עזיבת האורחים יש להחליף מצעים בכל אחת מהמיטות שבחדר, לשטוף את הריצפה ולנקות אבק בחדר ובמרפסת, להוציא מיטת תינוק בחדר משפחה (אם יש), לנקות את הג׳קוזי בסוויטה (אם יש) ולשאוב שטיחים בסוויטה.
הזמן הדרוש לכל אחד מסוגי עבודת ניקיון הוא:

- החלפת מצעים למיטה אחת – חמש דקות,
- שטיפת ריצפה וניקוי אבק – עשר דקות בחדר רגיל וחדר משפחה, עשרים דקות בסוויטה
- ניקוי מרפסת – חמש דקות
- קיפול והוצאת מיטת תינוק – חמש דקות
- ניקוי ג׳קוזי – עשרים דקות
- שאיבת שטיחים – חמש עשרה דקות

כתבו פעולה או פעולות לחישוב ולהחזרה של הזמן הדרוש לניקיון חדר.
יש לציין באיזו מחלקה צריך להוסיף את הפעולה או את הפעולות.

(1 נק׳) ד. למחלקה `Hotel` (מלון) יש תכונה אחת – מערך של חדרי המלון.

כתבו את כותרת המחלקה והגדירו את התכונה.

(6 נק׳) ה. בסוף כל יום עבודה האחראי על הניקיון מכין תוכנית העבודה ליום המחרת. לשם כך הוא מקבל רשימה של <u>מספרי חדרים</u> שיתפנו למחרת.

כתבו במחלקה `Hotel` פעולה פנימית המקבלת מערך של מספרים שלמים (מערך מספרי חדרים).

הפעולה תחשב ותחזיר את הזמן הכולל של הניקיון. הפעולה גם תדפיס את מספר מיטות התינוק שיתפנו.

כותרת הפעולה:

```
public int totalTime(int[] numArr)
```

שאלה 7

(5 נק') א. לפניכם פעולה רקורסיבית המקבלת הפנייה לשורש של עץ בינרי:

```
public static  boolean special(BinNode<Integer> t)
 {
     return special(t, 0);
 }
private static  boolean special(BinNode<Integer> t, int r)
 {
    if (t==null)
         return true;
    if (t.getValue()<r)
         return false;
   return special(t.getLeft(), r+1) && special(t.getRight(), r+1);
}
```

תנו דוגמה לעץ המכיל לפחות שישה איברים שעבורו תחזיר הפעולה `special(t)` את הערך `true`.

(4 נק') ב. מהי מטרת הפעולה `special`?

(4 נק') ג. נתונות שתי פעולות המקבלות הפנייה לשורש של **עץ חיפוש בינרי** (`Binary Search Tree`):

```
public static int what(BinNode<Integer> bt)
{
    if(bt.getLeft() == null)
        return bt.getValue();
    return what(bt.getLeft());
}

public static int where(BinNode<Integer> bt)
{
    while(bt.getRight()!=null)
    {
        bt = bt.getRight();
    }
    return bt.getValue();
}
```

מהן מטרות הפעולות `what` ו-`where`?

(5 נק') ד. האם קיים **עץ חיפוש בינרי** `bt` הכולל לפחות שלושה צמתים שעבורו מתקיים:

`what(bt) = where(bt)`?

אם כן – ציירו את העץ ולא – הסבירו מדוע עץ כזה לא קיים.

שאלה 8

נתונות שתי המחלקות הבאות:

```
public class A {
    public static int countA = 0;
    private String st;
    public A() { this.st = "aaa"; countA++;}
    public A(String st) { this.st = st; countA++;}
    public String toString() { return this.st; }
    public boolean same (Object obj) {
        if ((obj instanceof A) && (((A)obj).st.equals(this.st)))
            return true;
        return false;
    }
}    //end of class A

public class B extends A {
    public static int countB = 0;
    private int num;
    public B() {
        super();
        this.num = 1;
        countB++;
    }
    public B(int num) {
        super();
        this.num = num;
        countB++;
    }
    public B(int num, String st) {
        super(st);
        this.num = Math.abs(num);
        countB++;
    }
    public String toString() {
        String s = this.num +"/";
        for(int i = 0; i< this.num; i++)
            s+=super.toString();
        return s;
    }
    public boolean same (Object obj) {
        if ((obj instanceof B) && (((B)obj).num == this.num))
            return true;
        return false;
    }
} //end of class B
```

(4 נק') א. לפניכם קטע קוד מהתוכנית הראשית:

```
A a = new A();
B b = new B();
if (a.same (b))
    System.out.println (a);
else
    System.out.println (b);
if (b.same(a))
    System.out.println (b);
else
    System.out.println (a);
```

מה יהיה הפלט של קטע התוכנית? הסבירו את תשובתכם.

(14 נק') ב. נתונה המחלקה `Tester` הבאה:

```
public class Tester {
    public static void print (Object[] arr)
    {
        for (int i = 0; i < arr.length; i++)
        {
            if (arr[i] != null)
            {
                if (arr[i] instanceof B)
                    System.out.println("B: " + arr[i]);
                else if (arr[i] instanceof A)
                       System.out.println("A: " + arr[i]);
                     else
                       System.out.println("Unknown");
            }
            else
             System.out.println("NULL");
        }
    }
    public static void main (String[] args)
    {
        A[] ar1 = new A[7];
        ar1[0] = new B(-4);
        ar1[1] = new A("bbb");
        ar1[2] = new B(3,"ccc");
        ar1[3] = new B();
        ar1[4] = new B(2, "ddd");
        ar1[5] = new A();
        print(ar1);
        System.out.println(A.countA+ " "+ B.countB);
    }
} //end of class Tester
```

עקבו אחרי ביצוע תוכנית הראשית (main) ורשמו מה יהיה פלט התוכנית. יש להציג את התכונות של כל העצמים שנוצרו במהלך ביצוע התוכנית.

חלק ג'

ענו על <u>אחת</u> מבין השאלות 9-10 (ערך שאלה – 16 נקודות).

שאלה 9

(8 נק') א. נתונה הפעולה הרקורסיבית what המקבלת תור של מספרים q ומספר x :

```
public  static  int  what (Queue<Integer> q , int x)
{
  if ( q.isEmpty() || x != q.head() )
          return 0;
    x = q.remove();
    return   1 + what(q , x);
}
```

1. מה יחזיר זימון הפעולה what(q,5) עבור התור q: [**<u>5</u>**,5,5,3,3,2,8] (**<u>5</u>** בראש התור)?
   <u>יש להראות מעקב אחרי ביצוע הפעולה.</u>
2. אחרי זימון הפעולה what(q, x) עבור התור q לא ריק, התור q נשאר ללא שינוי.
   האם הטענה "ערך x לא מופיע בתור" נכונה? הסבירו את תשובתכם.
3. תנו דוגמה של תור q שעבורו זימון הפעולה what(q, 7) תחזיר 4. איך ייראה התור אחרי זימון הפעולה?
4. מה מבצעת הפעולה ?

(8 נק') ב. נתונה הפעולה secret המקבלת תור של מספרים q :

```
public  static  int  secret(Queue<Integer> q)
{
  if (q.isEmpty())  return 0;
  int x = q.head();
  int z = x * what(q, x);
  int y = 1 + secret(q);
  q.insert(z);
  return y;
}
```

עקבו אחר הפעולה secret(q) עבור התור q : [**<u>5</u>**,5,1,1,4,4,4,5,6] (5 בראש התור)
אין צורך לפרט את המעקב אחרי הפעולה what.

1. מהו הערך המוחזר מהפעולה secret?
2. איך ייראה התור בסיום הפעולה secret?
3. **<u>אחרי</u>** זימון הפעולה secret(q) התור q נראה כך: q:[**<u>6</u>**,16,2,3] (6 בראש התור)
   איך נראה התור q **<u>לפני</u>** זימון הפעולה?

שאלה 10

המחלקה Card מייצגת קלף למשחק ילדים Eka.

במשחק משתתפים קלפים בארבעה צבעים: אדום (R), ירוק (G), כחול (B) וצהוב (Y). על כל קלף רשום מספר בין 1 ל-9. סה"כ משתמשים במשחק 36 קלפים שונים.

```
public class Card
{
    private char color;
    private int digit;
    public Card(char c, int d)
    {
        this.color = c;
        this.digit = d;
    }
}
```

(3 נק') א. קלף c1 נחשב "חזק יותר" מקלף c2 אם המספר שרשום על הקלף c1 גדול מהמספר שרשום על הקלף c2. אם שני המספרים הם זהים יש להשוות צבעים לפי סדר הבא: אדום (הכי חלש), ירוק, כחול, צהוב (הכי חזק).

כתבו במחלקה Card פעולה compareTo. הפעולה תקבל הפניה לעצם other מטיפוס Card.

- אם הקלף שמפעיל את הפעולה (this) "חזק יותר" מהקלף שמתקבל כפרמטר (other), הפעולה תחזיר 1-.
- אם הקלף שמפעיל את הפעולה (this) "חלש יותר" מהקלף שמתקבל כפרמטר (other), הפעולה תחזיר 1.
- אם הקלף שמפעיל את הפעולה (this) זהה לקלף שמתקבל כפרמטר (other), הפעולה תחזיר 0.

כותרת הפעולה:

```
public int compareTo(Card other)
```

למשחק Eka כללים הבאים:

שני שחקנים לוקחים כל אחד קלף מראש החפיסה משותפת. שחקן שהקלף שלו "חזק יותר" מוסיף את זוג הקלפים לחפיסה שלו. אם אחד מהשחקנים צבר את כל הקלפים מצבע כלשהו, הוא ניצח במשחק. אם כל הקלפים חולקו ואין מנצח – מכריזים על תיקו.

המחלקה Deck מייצגת חפיסת קלפים. במחלקה הוגדרו פעולות הבאות:

`public Deck()`	פעולה בונה המייצרת חפיסה ריקה (ללא קלפים)
`public void addCard(Card c)`	פעולה המקבלת קלף c ומוסיפה אותו לחפיסה
`public Card drawCard()`	פעולה המוציאה קלף מהחפיסה ומחזירה אותו. אם החפיסה ריקה, הפעולה תחזיר null.
`public boolean exist(Card c)`	פעולה המקבלת קלף c ובודקת אם הוא קיים בחפיסה. אם כן – הפעולה תחזיר ערך true, ולא – הפעולה תחזיר ערך false.
`public boolean isFull(char color)`	פעולה המקבלת צבע color ובודקת אם בחפיסה יש כל הקלפים של אותו צבע. אם כן – הפעולה תחזיר ערך true, ולא – הפעולה תחזיר ערך false.

(5 נק') ב. כתבו פעולה חיצונית המקבלת **חפיסת קלפים מלאה** (הכוללת את כל 36 הקלפים) ומדמה את תהליך המשחק. הפעולה תדפיס הודעה על תוצאת המשחק – אחת מההודעות הבאות: "הראשון מנצח", "השני מנצח" או "תיקו".

כותרת הפעולה:

```
public static void game(Deck d)
```

(8 נק') ג. כתבו את המחלקה Deck: הגדירו תכונה או תכונות של המחלקה וממשו את כל פעולות שהוגדרו בה.

הערה: שימו לב שבמחלקה Card **אין פעולות get** ואי אפשר להוסיף אותן!

מבחן ב- C#

חלק א'

ענו על שלוש מבין השאלות 1-4 (ערך כל שאלה – 16 נקודות).

שאלה 1

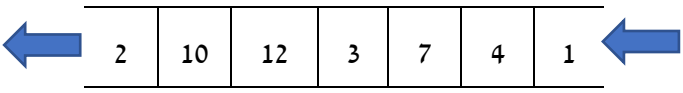
(6 נק') א. כתבו פעולה `PutInPlace` המקבלת תור `q` ומספר `num` וממקמת את המספר `num` במיקומו, כך שכל האיברים הקטנים ממנו, נמצאים לפניו וכל האיברים השווים לו או הגדולים ממנו נמצאים אחריו.

(אין משמעות לסדר של האיברים הקטנים או הגדולים ממנו).

הפעלה תחזיר את מיקומו הסידורי של המספר `num` בתוך התור אחרי הכנסתו.

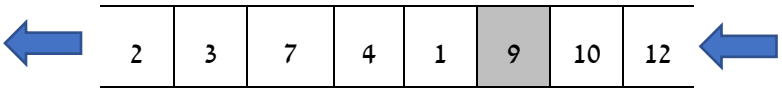
לדוגמא:

עבור התור q הבא:



והמספר num=9

התור שיתקבל יכול להיות



והפעולה תחזיר **6** (מיקומו של ה-**num** אחרי הכנסתו).

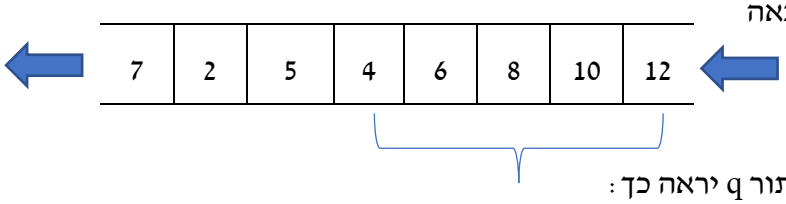
כותרת הפעולה:

```
public static int PutInPlace(Queue<int> q, int num)
```

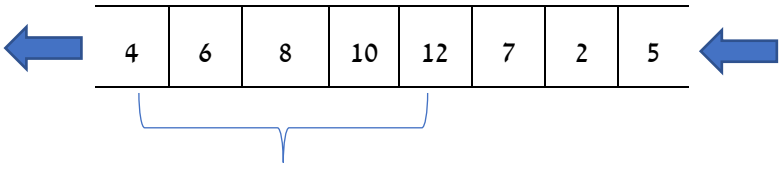
(6 נק') ב. כתבו פעולה חיצונית המקבלת תור `q` של מספרים שלמים ומספר שלם וחיובי `k`, ומעבירה את **האיברים האחרונים בתור אל ראש התור**. אפשר להניח ש-k קטן מאורך התור.

לדוגמה:

עבור התור q הבאה



והמספר k=5 התור q יראה כך:



כותרת הפעולה:

```
public static void MoveToFront(Queue<int> q, int k)
```

(4 נק') ג. מהי הסיבוכיות של הפעולות שכתבתם בסעיפים א' ו-ב'? **הסבירו את תשובתכם.**

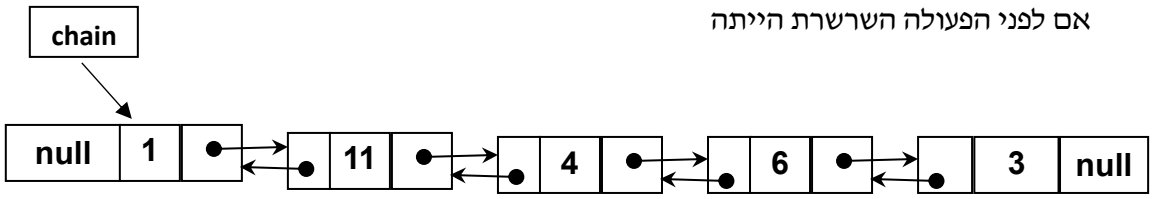
שאלה 2

בעזרת המחלקה `BinNode` אפשר לממש שרשרת חוליות דו-כיוונית.

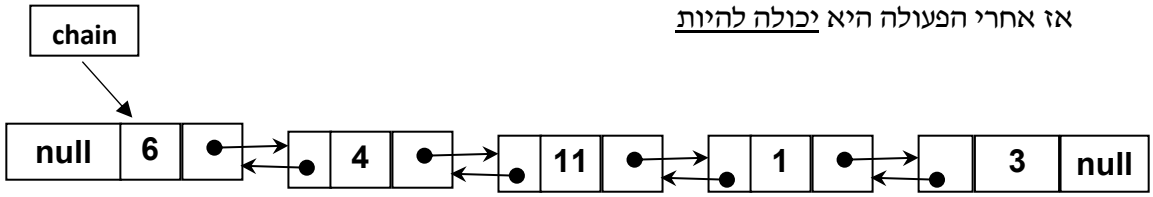
(12 נק׳) א. כתבו פעולה המקבלת הפנייה לחוליה הראשונה (הכי שמאלית) של שרשרת חוליות הדו-כיוונית ומסדרת אותה כך שכל הערכים הזוגיים יהיו בתחילת השרשרת וכל הערכים האי-זוגיים יהיו בסופה. סדר האיברים לא חשוב.

לדוגמה:

אם לפני הפעולה השרשרת הייתה



אז אחרי הפעולה היא <u>יכולה להיות</u>



שימו לב: אין להשתמש במבנה נתונים נוסף!

כותרת הפעולה:

```
public static void Order(BinNode<int> chain)
```

(4 נק׳) ב. מהי סיבוכיות הפעולה `Order` שכתבתם בסעיף א׳? **הסבירו את תשובתכם**.

שאלה 3

נתונות שתי המחלקות Flower ו- Rose:

```
public class Flower
{
  protected  int  height;
  private int price;
  public  Flower (int  val) { this.height = val; this.price = 10;}
  public  int  GetHeight()  { return this.height; }
  public  int  GetPrice ()  { return this.price; }
}
public class Rose : Flower
{
  private  string  color;
  public Rose(int val, string col): base(val)
    {
        this.color=col;
    }
  public  bool  ValidHeight()
    {
       return this.height > 10 && this.height < 30;
    }
}
```

(5 נק') א. לפניכם חמישה היגדים. קבעו לכל אחד מהם אם הוא נכון או אינו נכון, ונמקו את קביעתכם.

1. המחלקה **Flower** יורשת את הפעולה `ValidHeight()` מהמחלקה **Rose**.
2. המחלקה **Rose** יורשת את כל התכונות ואת כל הפעולות של המחלקה **Flower**.
3. המחלקה **Rose** יכולה לגשת ישירות לתכונה `height` של המחלקה **Flower**.
4. המחלקה **Flower** יכולה לגשת לתכונה `color` של המחלקה **Rose**.
5. לעצמים מטיפוס **Rose** אין תכונה `price`.

(5 נק') ב. לפניכם קטע קוד מהתוכנית הראשית:

```
Flower  first = new Rose (20, "RED");
Flower  second = new Flower (93);
```

בעבור כל אחת מההוראות שלפניך קבעו אם היא תקינה או אינה תקינה.

אם היא אינה תקינה, נמקו את קביעתכם וכתבו אם זו שגיאת ריצה או שגיאת הידור (קומפילציה).

```
1. bool b = first. ValidHeight();
2. bool b = second. ValidHeight();
3. bool b = ((Rose)first). ValidHeight();
4. bool b = ((Rose)second). ValidHeight();
5. bool b = first.GetPrice() == second.price;
```

(6 נק') ג. כתבו פעולה המקבלת מערך עצמים מטיפוס Object. על הפעולה להדפיס כמה עצמים הם מטיפוס **Rose**, כמה עצמים מטיפוס **Flower** ואינו מטיפוס **Rose** וכמה עצמים הם לא מטיפוס **Flower**.

שאלה 4

בפרויקט מסוים הוגדרו שלשה ממשקים ושתי מחלקות:

```
public interface Dancer{
    void LonelyDance();
    void CoupleDance(Dancer d);
}
public interface Singer  {    void Sing(); }
public interface Musician {   void Play(); }
public class Actor : Dancer, Singer, Musician
{
    private string name;
    private int age;
    …
}
public class Driver
{
    public static void Main(string[] args)
    {
        Actor actor1 = new Actor("Bob", 25);
        Actor actor2 = new Actor("Alice");
        ( *****)
    }
}
```

(4 נק') א. השלימו את המחלקה Actor: תכתבו כותרות של הפעולות החסרות בה.

(6 נק') ב. בכל אחד מסעיפים הבאים השורה (*****) תוחלף בשורת קוד משורות הקוד 1-6. כתבו, עבור כל אחד מהסעיפים, אם הקוד תקין או לא. אם הקוד אינו תקין – יש להסביר למה הוא אינו תקין ולציין את סוג השגיאה (קומפילציה או זמן ריצה).

שימו לב שאין קשר בין סעיפים!

```
(1)   Dancer dancer1 = actor1;
      dancer1. CoupleDance (actor2);

(2)   Singer singer2 = new Singer();
      singer2.Sing();

(3)   Singer singer3 = new Actor ("Clark",30);
      Dancer dancer3 = (Actor)singer3;

(4)   Singer actor4 = new Actor("Danny");
      actor4.Play();

(5)   Dancer actor5 = new Actor("Eddie");
      Singer singer5 = (Singer) actor5 ;
      singer5.Sing();

(6)   Musician musician6 = new Actor("Freddie", 45);
      ((Actor) musician6).LonelyDance();
      ((Singer) musician6).Sing();
```

(6 נק') ג. **(אין קשר לסעיפים א' ו-ב')**

נתונה הפעולה Main הבאה:

```
public static void Main(string[] args)
{
   C c = new A();
   B b1 = (B)(new A());
   B b2 = new D();
   A a = new D();
}
```

עבור כל אחת מן האפשרויות 1-6, קבעו אם יחס ההורשה בין המחלקות מתאים לקוד של הפעולה Main הנתונה.

הסבירו את תשובתכם.

1. C יורשת מ- A, B יורשת מ- A, D יורשת מ- A.
2. A יורשת מ- C, B יורשת מ- A, D יורשת מ- A.
3. C יורשת מ- A, B יורשת מ- A, D יורשת מ- B.
4. A יורשת מ- C, C יורשת מ- B, D יורשת מ- A.
5. A יורשת מ- C, B יורשת מ- A, D יורשת מ- B.
6. A יורשת מ- C, B יורשת מ- C, D יורשת מ- C.

חלק ב׳

ענו על <u>שתיים</u> 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 <u>פעם אחת</u> 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.**

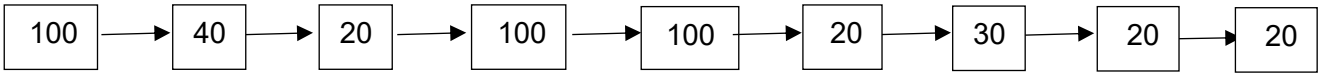
שימו לב: הפעולה <u>תעדכן</u> את הרשימה שהתקבלה כך שכל ערך יופיע פעם אחת ולאחריו מספר הפעמיים שהופיע ברשימה המקורית.

יש למחוק ערכים שחוזרים על עצמם חוץ מהמופע הראשון.

אין להשתמש במבני נתונים נוספים!!!

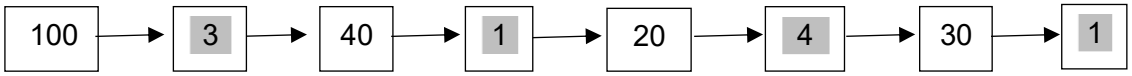
לדוגמה:

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<int> chain)
```

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

כותרת הפעולה:

```
public static int MostPopularNumber(Node<int> chain)
```

חובה להשתמש בפעולה שכתבתם בסעיף א׳.

(4 נק׳) ג. מהי סיבוכיות הפעולות BuildFreqList ו-MostPopularNumber מהסעיפים א׳ ו-ב׳?

הסבירו את תשובתכם.

שאלה 6

במלון ״בא ונח״ אפשר להזמין שלושה סוגים שונים של חדרים – חדר רגיל, חדר משפחתי וסוויטה.
מערכת ניהול המלון כוללת ארבע מחלקות: `Room`, `FamilyRoom`, `Suite`, `Hotel`.

לחדר רגיל (`Room`) יש את התכונות הבאות:

- `num` – מספר חדר, מטיפוס שלם, `int`
- `beds` – מספר המיטות, מטיפוס שלם, `int`
- `price` – מחיר, מטיפוס ממשי, `double`

לחדר משפחה (`FamilyRoom`) יש את התכונות הבאות:

- `num` – מספר חדר, מטיפוס שלם, `int`
- `beds` – מספר המיטות, מטיפוס שלם, `int`
- `price` – מחיר, מטיפוס ממשי, `double`
- `babyBed` – האם נוספה מיטת תינוק, מטיפוס `bool` (`true` - כן נוספה מיטה, `false` – לא)
- `balcony` – האם יש מרפסת, מטיפוס `bool` (`true` – יש, `false` - אין)

לסוויטה (`Suite`) יש את התכונות הבאות:

- `num` – מספר חדר, מטיפוס שלם, `int`
- `beds` – מספר המיטות, מטיפוס שלם, `int`
- `price` – מחיר, מטיפוס ממשי, `double`
- `jacuzzi` – האם יש ג׳קוזי, מטיפוס `bool` (`true` – יש, `false` – אין)

(2 נק׳) א. סרטטו תרשים UML המתאר את הקשר בין המחלקות: `Room`, `FamilyRoom`, `Suite`
באופן <u>המתאים ביותר לעקרונות של תכנות מונחה עצמים</u>.

(3 נק׳) ב. לכל אחת מהמחלקות `Room`, `FamilyRoom`, `Suite` כתבו כותרת המחלקה ותכונות שלה.
אפשר להניח שבכל מחלקה הוגדרו פעולות `Set/Get`.

(6 נק׳) ג. לאחר עזיבת האורחים יש להחליף מצעים בכל אחת מהמיטות שבחדר, לשטוף את הריצפה ולנקות אבק בחדר ובמרפסת, להוציא מיטת תינוק בחדר משפחה (אם יש), לנקות את הג׳קוזי בסוויטה (אם יש) ולשאוב שטיחים בסוויטה.
הזמן הדרוש לכל אחד מסוגי עבודת ניקיון הוא:

- החלפת מצעים למיטה אחת – חמש דקות.
- שטיפת ריצפה וניקוי אבק – עשר דקות בחדר רגיל וחדר משפחה, עשרים דקות בסוויטה.
- ניקוי מרפסת – חמש דקות.
- קיפול והוצאת מיטת תינוק – חמש דקות.
- ניקוי ג׳קוזי – עשרים דקות.
- שאיבת שטיחים – חמש עשרה דקות.

כתבו פעולה או פעולות לחישוב ולהחזרה של הזמן הדרוש לניקיון חדר.
יש לציין באיזו מחלקה צריך להוסיף את הפעולה או את הפעולות.

(1 נק׳) ד. למחלקה `Hotel` (מלון) יש תכונה אחת – מערך של חדרי המלון.

כתבו את כותרת המחלקה והגדירו את התכונה.

(6 נק׳) ה. בסוף כל יום עבודה האחראי על הניקיון מכין תוכנית העבודה ליום המחרת. לשם כך הוא מקבל רשימה של <u>מספרי חדרים</u> שיתפנו למחרת.

כתבו במחלקה `Hotel` פעולה פנימית המקבלת מערך של מספרים שלמים (מערך מספרי חדרים).

הפעולה תחשב ותחזיר את הזמן הכולל של הניקיון. הפעולה גם תדפיס את מספר מיטות התינוק שיתפנו.

כותרת הפעולה:

```
public int TotalTime(int[] numArr)
```

שאלה 7

(5 נק') א. לפניכם פעולה רקורסיבית המקבלת הפנייה לשורש של עץ בינרי:

```
public static  bool Special(BinNode<int> t)
 {
     return Special(t, 0);
 }
private static  bool Special(BinNode<int> t, int r)
 {
    if (t==null)
         return true;
    if (t.GetValue()<r)
         return false;
   return Special(t.GetLeft(), r+1) && Special(t.GetRight(), r+1);
}
```

תנו דוגמה לעץ המכיל לפחות שישה איברים שעבורו תחזיר הפעולה `Special(t)` את הערך `true`.

(4 נק') ב. מהי מטרת הפעולה `Special`?

(4 נק') ג. נתונות שתי פעולות המקבלות הפנייה לשורש של **עץ חיפוש בינרי** (`Binary Search Tree`):

```
public static int What(BinNode<int> bt)
{
    if(bt.GetLeft() == null)
        return bt.GetValue();
    return What(bt.GetLeft());
}

public static int Where(BinNode<int> bt)
{
    while(bt.GetRight()!=null)
    {
        bt = bt.GetRight();
    }
    return bt.GetValue();
}
```

מהן מטרות הפעולות `What` ו-`Where`?

(5 נק') ד. האם קיים **עץ חיפוש בינרי** `bt` הכולל לפחות שלושה צמתים שעבורו מתקיים:

`What(bt) = Where(bt)`?

אם כן – ציירו את העץ ולא – הסבירו מדוע עץ כזה לא קיים.

שאלה 8

נתונות שתי המחלקות הבאות:

```
public class A {
    public static int countA = 0;
    private string st;
    public A() { this.st = "aaa"; countA++;}
    public A(string st) { this.st = st; countA++;}
    public override string ToString() { return this.st; }
    public virtual bool Same (Object obj) {
        if ((obj is A) && (((A)obj).st.Equals(this.st)))
            return true;
        return false;
    }
}    //end of class A

public class B : A {
    public static int countB = 0;
    private int num;
    public B(): base() {
        this.num = 1;
        countB++;
    }
    public B(int num): base() {
        this.num = num;
        countB++;
    }
    public B(int num, string st): base(st) {
        this.num = Math.Abs(num);
        countB++;
    }
    public override string ToString() {
        string s = this.num +"/";
        for(int i = 0; i< this.num; i++)
            s+= base.ToString();
        return s;
    }
    public override bool Same (Object obj) {
        if ((obj is B) && (((B)obj).num == this.num))
            return true;
        return false;
    }
} //end of class B
```

(4 נק') א. לפניכם קטע קוד מהתוכנית הראשית:

```
A a = new A();
B b = new B();
if (a.Same(b))
    Console.WriteLine (a);
else
    Console.WriteLine (b);
if (b.Same(a))
    Console.WriteLine (b);
else
    Console.WriteLine (a);
```

מה יהיה הפלט של קטע התוכנית? הסבירו את תשובתכם.

(14 נק') ב. נתונה המחלקה `Tester` הבאה:

```
public class Tester {
    public static void Print (Object[] arr)
    {
        for (int i = 0; i < arr.Length; i++)
        {
            if (arr[i] != null)
            {
                if (arr[i] is B)
                    Console.WriteLine("B: " + arr[i]);
                else if (arr[i] is A)
                      Console.WriteLine("A: " + arr[i]);
                     else
                      Console.WriteLine("Unknown");
            }
            else
             Console.WriteLine("NULL");
        }
    }
    public static void Main (String[] args)
    {
        A[] ar1 = new A[7];
        ar1[0] = new B(-4);
        ar1[1] = new A("bbb");
        ar1[2] = new B(3,"ccc");
        ar1[3] = new B();
        ar1[4] = new B(2, "ddd");
        ar1[5] = new A();
        Print(ar1);
        Console.WriteLine(A.countA+ " "+ B.countB);
    }
} //end of class Tester
```

עקבו אחרי ביצוע תוכנית הראשית (main) ורשמו מה יהיה פלט התוכנית. יש להציג את התכונות של כל העצמים שנוצרו במהלך ביצוע התוכנית.

חלק ג'

ענו על אחת מבין השאלות 9-10 (ערך שאלה – 16 נקודות).

שאלה 9

(8 נק') א. נתונה הפעולה הרקורסיבית What המקבלת תור של מספרים q ומספר x :

```
public  static  int  What (Queue<int> q , int x)
{
  if ( q.IsEmpty() || x != q.Head() )
          return 0;
    x = q.Remove();
    return   1 + What(q , x);
}
```

1. מה יחזיר זימון הפעולה What(q,5) עבור התור q: [**5**,5,5,3,3,2,8] **(5 בראש התור)**?
   <u>יש להראות מעקב אחרי ביצוע הפעולה</u>.
2. אחרי זימון הפעולה What(q, x) עבור התור q לא ריק, התור q נשאר ללא שינוי.
   האם הטענה "ערך x לא מופיע בתור" נכונה? הסבירו את תשובתכם.
3. תנו דוגמה של תור q שעבורו זימון הפעולה What(q, 7) תחזיר 4. איך ייראה התור אחרי זימון הפעולה?
4. מה מבצעת הפעולה ?

(8 נק') ב. נתונה הפעולה Secret המקבלת תור של מספרים q :

```
public  static  int  Secret(Queue<int> q)
{
  if (q.isEmpty())  return 0;
  int x = q.Head();
  int z = x * What(q, x);
  int y = 1 + Secret(q);
  q.Insert(z);
  return y;
}
```

עקבו אחר הפעולה Secret(q) עבור התור q : [**5**,5,1,1,4,4,4,5,6] **(5 בראש התור)**.
אין צורך לפרט את המעקב אחרי הפעולה What.

1. מהו הערך המוחזר מהפעולה Secret?
2. איך ייראה התור בסיום הפעולה Secret?
3. **<u>אחרי</u>** זימון הפעולה Secret(q) התור q נראה כך: q: [**6**,16,2,3] **(6 בראש התור)**
   איך נראה התור q **<u>לפני</u>** זימון הפעולה?

שאלה 10

המחלקה Card מייצגת קלף למשחק ילדים Eka. במשחק משתתפים קלפים בארבעה צבעים: אדום (R), ירוק (G), כחול (B) וצהוב (Y). על כל קלף רשום מספר בין 1 ל-9. סה"כ משתמשים במשחק 36 קלפים שונים.

```
public class Card
{
    private char color;
    private int digit;
    public Card(char c, int d)
    {
        this.color = c;
        this.digit = d;
    }
}
```

(3 נק') א. קלף c1 נחשב "חזק יותר" מקלף c2 אם המספר שרשום על הקלף c1 גדול מהמספר שרשום על הקלף c2. אם שני המספרים הם זהים יש להשוות צבעים לפי סדר הבא: אדום (הכי חלש), ירוק, כחול, צהוב (הכי חזק).

כתבו במחלקה Card פעולה CompareTo. הפעולה תקבל הפניה לעצם other מטיפוס Card.

- אם הקלף שמפעיל את הפעולה (this) "חזק יותר" מהקלף שמתקבל כפרמטר (other), הפעולה תחזיר 1-.
- אם הקלף שמפעיל את הפעולה (this) "חלש יותר" מהקלף שמתקבל כפרמטר (other), הפעולה תחזיר 1.
- אם הקלף שמפעיל את הפעולה (this) זהה לקלף שמתקבל כפרמטר (other), הפעולה תחזיר 0.

כותרת הפעולה:

```
public int CompareTo(Card other)
```

למשחק Eka כללים הבאים:

שני שחקנים לוקחים כל אחד קלף מראש החפיסה משותפת. שחקן שהקלף שלו "חזק יותר" מוסיף את זוג הקלפים לחפיסה שלו. אם אחד מהשחקנים צבר את כל הקלפים מצבע כלשהו, הוא ניצח במשחק. אם כל הקלפים חולקו ואין מנצח – מכריזים על תיקו.

המחלקה Deck מייצגת חפיסת קלפים. במחלקה הוגדרו פעולות הבאות:

פעולה בונה המייצרת חפיסה ריקה (ללא קלפים).	`public Deck()`
פעולה המקבלת קלף c ומוסיפה אותו לחפיסה.	`public void AddCard(Card c)`
פעולה המוציאה קלף מהחפיסה ומחזירה אותו. אם החפיסה ריקה, הפעולה תחזיר null.	`public Card DrawCard()`
פעולה המקבלת קלף c ובודקת אם הוא קיים בחפיסה. אם כן – הפעולה תחזיר ערך true, ולא – הפעולה תחזיר ערך false.	`public bool Exist(Card c)`
פעולה המקבלת צבע color ובודקת אם בחפיסה יש כל הקלפים של אותו צבע. אם כן – הפעולה תחזיר ערך true, ולא – הפעולה תחזיר ערך false.	`public bool IsFull(char color)`

(5 נק׳) ב. כתבו פעולה חיצונית המקבלת **חפיסת קלפים מלאה** (הכוללת את כל 36 הקלפים) ומדמה את תהליך המשחק. הפעולה תדפיס הודעה על תוצאת המשחק – אחת מההודעות הבאות: ״הראשון מנצח״, ״השני מנצח״ או ״תיקו״.

כותרת הפעולה:

```
public static void Game(Deck d)
```

(8 נק׳) ג. כתבו את המחלקה Deck: הגדירו תכונה או תכונות של המחלקה וממשו את כל פעולות שהוגדרו בה.

הערה: שימו לב שבמחלקה Card **אין פעולות Get** ואי אפשר להוסיף אותן!

בהצלחה!

מילון עזר – בחינת מה"ט 97105 מועד א קיץ 23 – תשפ"ג

חלק א القسم أ

רקם השאלה	הכלמה \ הביטוי בעברית	الكلمة \ التعبير بالعربية
1	אין צורך במילון	لا توجد حاجة للقاموس
2	אין צורך במילון	لا توجد حاجة للقاموس
3	היגד	مقولة
4	אין צורך במילון	لا توجد حاجة للقاموس

חלק ב القسم ب

רקם השאלה	הכלמה \ הביטוי בעברית	الكلمة \ التعبير بالعربية
5	אין צורך במילון	لا توجد حاجة للقاموس
6	סוויטה(חדר מפואר במלון)	جناح (غرفة فندقية فاخرة)
6	מיטה \ מיטות	سرير \ اسرة
6	מיטת תינוק	سرير طفل
6	אורחים	ضيوف
6	מצעים	بياضات (أغطية السرير)
6	לשטוף ריצפה	غسل الأرضية
6	ניקוי אבק	تنظيف الغبار
6	שאיבת שטיחים	تنظيف السجاد بالمكنسة الكهربائية
6	קיפול	طي
6	ניקוי מרפסת	تنظيف الشرفة
7	אין צורך במילון	لا توجد حاجة للقاموس
8	אין צורך במילון	لا توجد حاجة للقاموس

חלק ג' القسم ج

رقم السؤال	الكلمة \ التعبير بالعبرية	الكلمة \ التعبير بالعربية
9	אין צורך במילון	لا توجد حاجة للقاموس
10	"חלש יותר"	"أضعف"
10	"חזק יותר"	"أقوى"
10	אדום	أحمر
10	ירוק	أخضر
10	כחול	أزرق
10	צהוב	أصفر
10	חפיסת קלפים	رزمة البطاقات (الشدة)
10	מדמה את תהליך המשחק	يُحاكي مجرى اللعبة
10	מנצח	فائز

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.

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

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.

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

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.

ממשק המחלקה הגנרית – חוליה בינרית<T>BinNode

המחלקה מגדירה חוליה בינרית שבה ערך מטיפוס T ושתי הפניות לחוליות בינריות.

הפעולה בונה חוליה בינרית. ערך החוליה הוא x וערך שתי ההפניות שלה הוא **null**	**BinNode** (T x)
הפעולה בונה חוליה בינרית שערכה יהיה x. left ו-right הן (הפניות אל) הילד השמאלי והימני שלה. ערכי ההפניות יכולים להיות **null**	BinNode (BinNode<T> left, T x, BinNode<T> right)
הפעולה מחזירה את הערך של החוליה	**T** getValue()
הפעולה משנה את הערך השמור בחוליה ל-x	**void** setValue (T x)
הפעולה מחזירה **true** אם יש ילד שמאלי	**boolean** hasLeft()
הפעולה מחזירה **true** אם יש ילד ימני	**boolean** hasRight()
הפעולה מחזירה את הילד השמאלי של החוליה. אם אין ילד שמאלי הפעולה מחזירה **null**	**BinNode<T>** getLeft()
הפעולה מחזירה את הילד הימני של החוליה. אם אין ילד ימני הפעולה מחזירה **null**	**BinNode<T>** getRight()
הפעולה מחליפה את הילד השמאלי בחוליה left	**void** setLeft (BinNode<T> left)
הפעולה מחליפה את הילד הימני בחוליה right	**void** setRight (BinNode<T> right)
הפעולה מחזירה מחרוזת המתארת את הערך השמור בחוליה	**String** toString()

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

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.

ממשק המחלקה חוליה בינרית BinNode <T>

המחלקה מגדירה חוליה בינרית שבה ערך מטיפוס T ושתי הפניות לחוליות בינריות.

הפעולה בונה חוליה בינרית. ערך החוליה הוא x וערך שתי ההפניות שלה הוא **null**	BinNode (**T** x)
הפעולה בונה חוליה בינרית שערכה יהיה x, left ו-right הן הפניות אל הילד השמאלי והימני שלה. ערכי ההפניות יכולים להיות **null**	BinNode (BinNode<**T**> left, T x, BinNode<T> right)
הפעולה מחזירה את הערך של החוליה	**T** GetValue()
הפעולה משנה את הערך השמור בחוליה ל-x	**void** SetValue (**T** x)
הפעולה מחזירה **true** אם יש ילד שמאלי	**bool** HasLeft()
הפעולה מחזירה **true** אם יש ימני	**bool** HasRight()
הפעולה מחזירה את הילד השמאלי של החוליה. אם אין ילד שמאלי הפעולה מחזירה **null**	**BinNode<T>** GetLeft()
הפעולה מחזירה את הילד הימני של החוליה. אם אין ילד ימני הפעולה מחזירה **null**	**BinNode<T>** GetRight()
הפעולה מחליפה את הילד השמאלי בחוליה left	**void** SetLeft (BinNode<**T**> left)
הפעולה מחליפה את הילד הימני בחוליה right	**void** SetRight (BinNode<**T**> right)
הפעולה מחזירה מחרוזת המתארת את הערך השמור בחוליה	**string** ToString()

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