משרד העבודה
MINISTRY OF LABOR

מה"ט
המכון הממשלתי להכשרה טכנולוגית

מספר ת"ז_____
מספר מחברת_____

מבני נתונים ותכנות מונחה עצמים

הנדסאים וטכנאים – הנדסת תוכנה

הנחיות לבחינה

מועד הבחינה: קיץ תשפ"ה – 2025 – מועד א'
מספר השאלון: 97105
נספח ממשקים לבחינה JAVA
נספח ממשקים לבחינה C#
מילון עזר

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-C# (עמוד 2)

מבחן ב-JAVA (עמוד 14)

בכל מבחן עשר שאלות.

חלק א' – 32 נקודות

שאלות 1–3: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 16 נקודות.

חלק ב' – 32 נקודות

שאלות 4–6: יש לענות על **שתי** שאלות בלבד. ערך כל שאלה 16 נקודות.

חלק ג' – 36 נקודות

שאלות 7–9: יש לענות על **שתי** שאלות בלבד. ערך השאלה 18 נקודות.

בסך הכול: 100 נקודות.

ג. חומר עזר מותר לשימוש:

1. מחשבון (אין להשתמש במחשב כף יד או במחשבון עם תקשורת חיצונית).
2. קלסר **אחד בלבד** עם **חומר ההרצאות**. 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 המבחן, ולוודא שהן מובנות.
2. את התשובות יש לכתוב בצורה מסודרת, בכתב יד ברור ונקי (גם בכך תלויה הערכת המבחן).
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!

מבחן ב-C#

הנחיות כלליות לנבחנים:

בנספחי הבחינה מובאים ממשקים (interface) הכוללים פונקציות ומחלקות עזר, יש להקפיד ולהשתמש בהן בשאלות הרלוונטיות בבחינה על-מנת להימנע מאיבוד ניקוד.

מבחן ב-C#

חלק א'

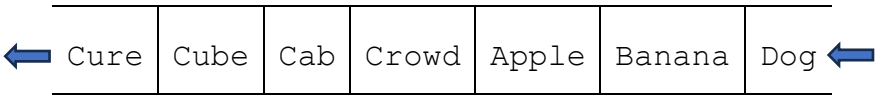
ענו על שתיים מבין השאלות 1–3 (ערך כל שאלה – 16 נקודות).

<u>שאלה 1</u>

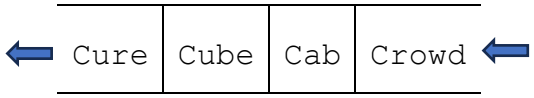
הגדרה:

תור מחרוזות "תקין" הוא תור שבו כל המחרוזות המתחילות באותה האות נמצאות רק בתחילת התור.

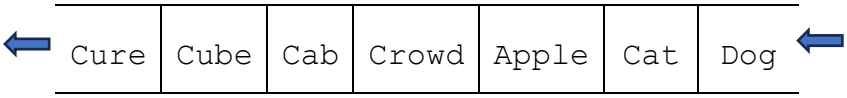
לדוגמה, התור שלפניכם הוא "תור תקין":



גם התור הזה הוא "תור תקין":



אבל התור הזה אינו "תור תקין":



(3 נק') א. כתבו פעולה המקבלת תור של מחרוזות "**תקין**" `q` ומחרוזת `st`. הפעולה תוסיף את המחרוזת לתור כך שהוא יישאר תקין.

כותרת הפעולה:

```
public static void AddToProperQueue(Queue<string> q, string s)
```

(5 נק') ב. כתבו פעולה המקבלת תור של מחרוזות. הפעולה תבדוק אם תור הוא "תור תקין". אם כן – הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

כותרת הפעולה:

```
public static bool IsProperQueue(Queue<string> q)
```

(5 נק') ג. כתבו פעולה המקבלת תור של מחרוזות. הפעולה תבדוק אם התור הוא "תור תקין". אם כן – הפעולה לא תבצע דבר, ואם לא – הפעולה תשנה את סדר המחרוזות בתור כך שהוא יהיה "**תקין**".

כותרת הפעולה:

```
public static void FixIt(Queue<string> q)
```

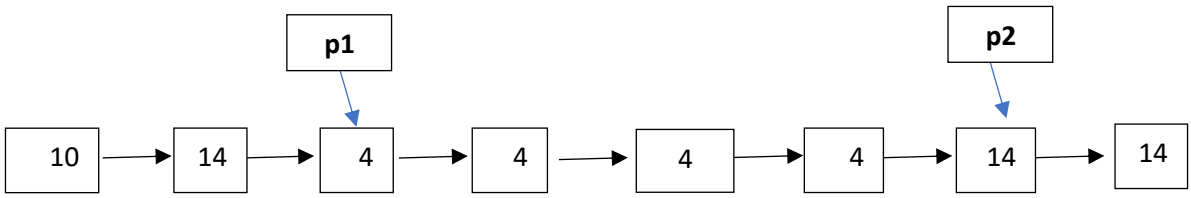
(3 נק') ד. מהי סיבוכיות הפעולות שכתבתם בסעיפים א'–ג'? **הסבירו את תשובתכם.**

<u>שאלה 2</u>

נגדיר "רצף K" כרצף של K חוליות של מספרים שלמים שהערך של כל אחת מהן הוא K.

(6 נק') א. כתבו פעולה המקבלת שתי הפניות p1 ו-p2 לחוליות שונות בשרשרת חוליות של מספרים שלמים. הפעולה תבדוק אם החוליות בין p1 (כולל) ל-p2 (לא כולל) מהוות את "רצף K". אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור השרשרת שלפניכם הפעולה תחזיר ערך true:



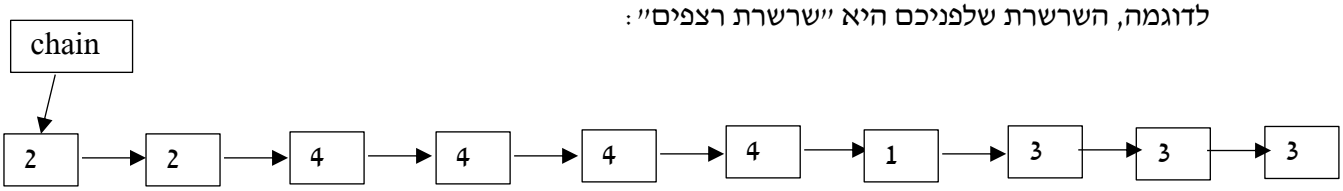
כותרת הפעולה:

```
public static bool IsSequenceK(Node<int> p1, Node<int> p2)
```

(8 נק') ב. שרשרת חוליות נקראת "שרשרת רצפים" אם היא מורכבת מכמה "רצפי K".

כתבו פעולה המקבלת הפניה לחוליה הראשונה של שרשרת חוליות של מספרים שלמים. הפעולה תבדוק אם השרשרת היא "שרשרת רצפים". אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, השרשרת שלפניכם היא "שרשרת רצפים":



כותרת הפעולה:

```
public static bool IsSequenceList(Node<int> chain)
```

(2 נק') ג. מהי סיבוכיות הפעולות שכתבתם בסעיפים א'–ב'? **הסבירו את תשובתכם.**

שאלה 3

בשאלה זו שלושה סעיפים. אין קשר בין הסעיפים.

(4 נק׳) א. נתונה פעולה ראשית במחלקה `Test`.

1. שרטטו את **עצי הירושה עבור שתי** אפשרויות היררכיה בין המחלקות כך שהפעולה הראשית תהיה תקינה והתוכנית תרוץ ללא שגיאות.

```
public class Test {
    public static void Main(string[] args){
        S s1 = new R();
        P p1 = (P)(new R());
        S s2 = new P();
        P p2 = new Q();
//      (*****)
    }
}
```

2. רוצים להוסיף בסוף התוכנית את שורת הקוד שלפניכם:

```
Q q1 =(Q) p1;
```

האם התוכנית הראשית תהיה תקינה עבור שתי האפשרויות שציינתם? אם לא – מהו סוג השגיאה שתתרחש?

(6 נק׳) ב. לפניכם המחלקות `First`, `Second` ו-`Program`. במחלקות `First` ו-`Second` **חסר** מימוש בנאים ופעולות `ToString`.

```
public class First {
 private static int c = 0;
 protected int num;
 public First(int num){…}
 public override string
            ToString(){…}
}
```

```
public class Second : First{
 private char ch;
 public Second(int num, char ch):
        base(num){…}
 public override string
        ToString(){…}
}
```

השלימו את הפעולות במחלקות `First` ו-`Second` כך שאחרי הרצת הפעולה הראשית של המחלקה `Program` יתקבל הפלט הזה:

```
public class Program{
 public static void Main(string[] args) {
  First f1 = new First(10);
  Console.WriteLine("f1 is "+f1);
  First f2 = new Second(3,'A');
  Console.WriteLine("f2 is "+f2);
  First f3 = new Second(4,'Z');
  Console.WriteLine("f3 is "+f3);
  First f4 = new First(7);
  Console.WriteLine("f4 is "+f4);
 }
}
```

הפלט שהתקבל אחרי הרצת התוכנית:

```
f1 is 1-10
f2 is 2-6-A
f3 is 3-12-Z
f4 is 4-28
```

(6 נק׳) ג. לפניכם קוד המחלקה `Two`. המחלקה יורשת מהמחלקה `One`.

```
public class Two : One
{
    private int z;
    public Two()
    {
        this.z = 11;
    }

    public Two(string s1, int z): base(s1)
    {
        this.z = z;
    }

    public override double P1(int a)
    {
        return a * base.P1(a);
    }

    public int P2()
    {
        return One.w + this.M();
    }

    public void Print()
    {
        One.D();
        Console.WriteLine(s+" SON");
    }
}
```

כתבו במחלקה `One` את התכונות וה**כותרות** של הפעולות הנדרשות.

יש לציין הרשאת הגישה (`public`, `protected`, `private`) לכל תכונה ולכל פעולה.

אין צורך לממש את הפעולות.

אין לשנות את הקוד של המחלקה `Two`.

חלק ב׳

ענו על <u>שתיים</u> מבין השאלות 4–6 (ערך כל שאלה – 16 נקודות).

<u>שאלה 4</u>

החברה ״סוד זה הצלחה״ החליטה שכל הישיבות והכנסים שלה יתקיימו באופן וירטואלי דרך מערכת מפגשים וירטואליים Boom שפיתחו מתכנתים שלה. במפגשי Boom ניתן לחלק משתתפים בין ״חדרים וירטואליים״. לשם ניהול המפגשים הווירטואליים פותחו מחלקות Meeting, VRoom, Participant.

- כל משתתף (Participant) מאופיין על ידי שם (name) ומספר היחידה שבה הוא עובד (dep, מספר שלם בין 1 ל-10).
- כל חדר (VRoom) מאופיין על ידי מספר חדר (num, מספר שלם וחיובי) ותור משתתפים (parts).
- למחלקה Meeting תכונה אחת – רשימת חדרים וירטואליים (rooms - שרשרת חוליות מסוג VRoom).

```
public class User {
 private string name;
 private int dep;
 public User(string n, int d){
  this.name = n;
  this.dep = d;
 }
}      //אפשר להניח שהוגדרו Set/Get
```

```
public class Meeting {
 private Node<VRoom> rooms;
 public Meeting() {
  this.rooms = null;
 }
}
```

```
public class Vroom {
 private int num;
 Queue<Participant> parts;
 public VRoom(int n){
  …
 }
}
```

(2 נק׳) א. כתבו במחלקה VRoom פעולה בונה המקבלת מספר חדר ובונה חדר ריק ממשתתפים.

(2 נק׳) ב. כתבו במחלקה VRoom פעולה add המקבלת משתתף ומוסיפה אותו לחדר.

(6 נק׳) ג. לפני כל ישיבה, הנהלת החברה מחליטה על מספר המשתתפים שיהיו בחדר וירטואלי.

כתבו במחלקה Meeting פעולה המקבלת מספר שלם וחיובי number. הפעולה תקלוט את פרטי המשתתפים (שם ומספר יחידה). הפעולה תיצור חדר שמספרו אחד ואליו יצטרפו number משתתפים ראשונים, לאחריו חדר מספר שתיים ואליו יצטרפו number משתתפים הבאים וכך הלאה. הקלט יסתיים כאשר ייקלט 0 כמספר המחלקה. כותרת הפעולה:

```
public void Divide (int number)
```

(6 נק׳) ד. כאשר מטרת הישיבה היא סיעור מוחות (brainstorming), חשוב שבכל חדר יהיו נציגים מכל עשר יחידות החברה.

כתבו במחלקה Meeting פעולה אשר מדפיסה עבור כל חדר את מספרי יחידות החברה שנציגיהן לא נמצאים בחדר. כותרת הפעולה:

```
public void PrintMissingUnits()
```

שאלה 5

(2 נק׳) א. כתבו פעולה **GetLast** המקבלת הפניה לחוליה הראשונה של שרשרת של מספרים שלמים.
הפעולה תחזיר הפנייה לחוליה האחרונה של השרשרת.

נתונה הפעולה **What** המקבלת שני מספרים שלמים **k<n**. הפעולה משתמשת בפעולה GetLast.

```
public static Node<int> What (int n, int k){
     if (n == k)
         return new Node<int>(n);
     else
     {
         if (k % 2 == 0)
             return new Node<int>(k, What(n, k+1));
         else
         {
             Node<int> p = new Node<int>(k);
             Node<int> chain = What (n, k+1);
             Node<int> last = GetLast (chain);
             last.SetNext (p);
             return chain;
          }
     }
 }
```

(6 נק׳) ב. עקבו אחרי זימון הפעולה What(10,5) ורשמו מה תחזיר הפעולה.

יש להראות את תוכן השרשרת בחזרה מכל קריאה הרקורסיבית.

נתונה הפעולה **Secret**:

```
public static Node<int> Secret (int n, int k) {
     if (n == k)
         return new Node<int>(n);
     else
     {
         if (n % 2 == 0)
             return new Node<int>(n, Secret(n-1, k));
         else
         {
             Node<int> chain = Secret (n-1, k);
             GetLast (chain).SetNext (new Node<int>(n));
             return chain;
         }
     }
 }
```

(3 נק׳) ג. מה תהיה תוצאת הזימון Secret(6, 2)?

(2 נק׳) ד. האם קיימים שני מספרים שלמים וחיוביים k<n כך שתוצאות הזימונים Secret(n, k) ו-What(n, k) יהיו זהות? אם כן – תנו דוגמה לזוג מספרים שכזה, ואם לא – הסבירו למה הדבר אינו אפשרי.

(3 נק׳) ה. כתבו את הפעולה What בצורה הלא רקורסיבית.

שאלה 6

(4 נק') א. כתבו פעולה `SumLeaves` שמקבלת עץ בינארי של מספרים שלמים ומחזירה את סכום הערכים של **כל העלים** בעץ.

נתונה המחלקה `NodeInfo` שמכילה שתי תכונות: מספר שלם `num`, ומחרוזת `info`.
במחלקה הוגדרה פעולה בונה:

```
public NodeInfo(int num, string info)
```

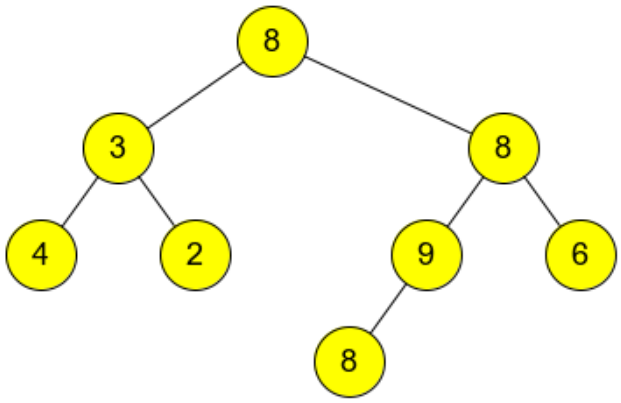
במחלקה הוגדרו פעולות `Get` ו-`Set` במחלקה `NodeInfo`.

(6 נק') ב. כתבו פעולה `CreateQueue` המקבלת עץ בינארי של מספרים שלמים ומחזירה תור מטיפוס `NodeInfo` שבו:

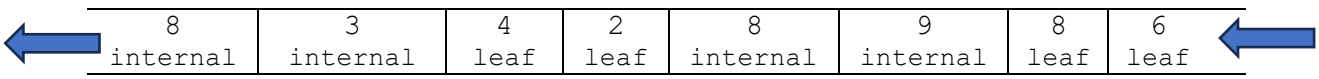
- ערך ה-`num` בכל איבר זהה לערך הצומת בעץ המתקבל כפרמטר.
- ערך ה-`info` מכיל את המחרוזת "*leaf*" אם הצומת הוא עלה, ואת המחרוזת "*internal*" אם הצומת הוא צומת פנימי.

סדר האיברים בתור לא חשוב!

לדוגמה, עבור העץ שלפניכם:



הפעולה תחזיר את תור הזה:



8 internal	3 internal	4 leaf	2 leaf	8 internal	9 internal	8 leaf	6 leaf

כותרת הפעולה:

```
public static Queue<NodeInfo> CreateQueue (BinNode<int> bt)
```

(6 נק') ג. כתבו פעולה `MaxLeaveValue` המקבלת עץ בינארי של מספרים שלמים ומחזירה את **הערך הגדול ביותר מבין העלים של העץ.**

חלק ג'

ענו על <u>שתיים</u> מבין השאלות 7–9 (ערך שאלה – 18 נקודות).

<u>שאלה 7</u>

(6 נק') א. כתבו פעולה המקבלת שני תורים של מספרים שלמים `q1` ו-`q2`. הפעולה תבדוק אם כל הערכים של תור `q1` קטנים מכל הערכים של `q2`. אם כן, הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

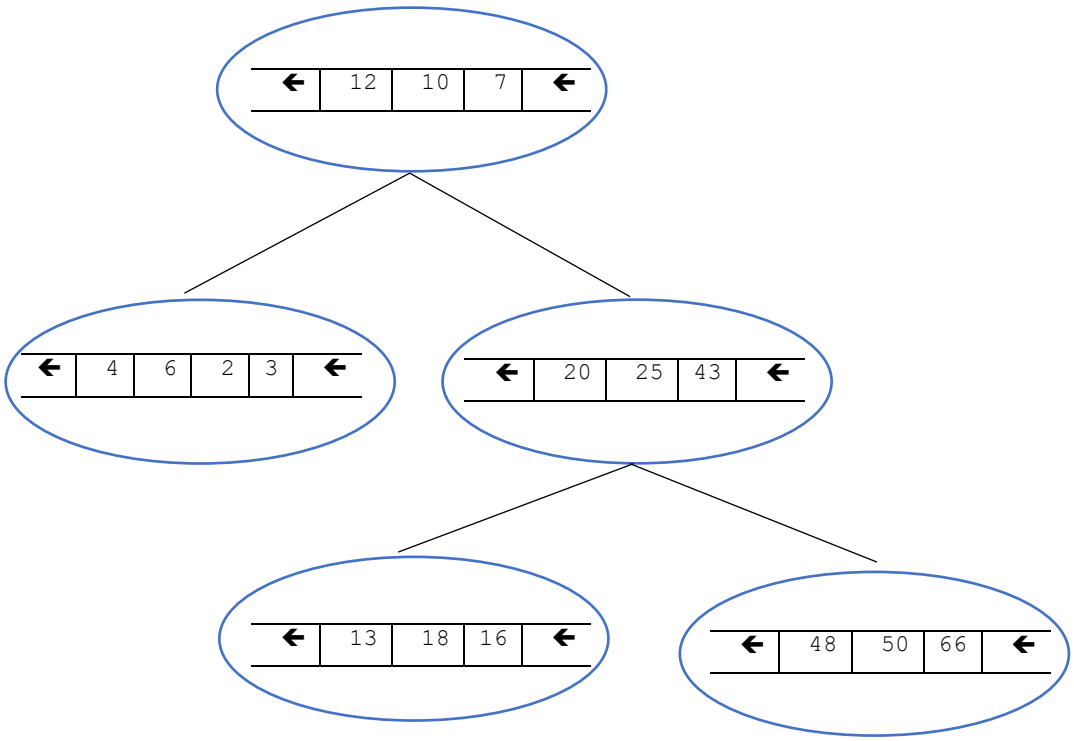
כותרת הפעולה:

```
public static bool IsSmaller(Queue<int>q1, Queue<int> q2)
```

(6 נק') ב. נתון עץ שבו כל צומת מכיל תור של מספרים שלמים. עץ נקרא "עץ מעולה" אם הוא עונה על שני התנאים האלה:

- לכל צומת שאינו עלה יש שני בנים.
- כל הערכים הנמצאים בצומת האב גדולים מכל הערכים הנמצאים בבן השמאלי וקטנים מכל הערכים הנמצאים בבן הימני.

לדוגמה, העץ שלפניכם הוא "עץ מעולה":



כתבו פעולה המקבלת עץ תורים של מספרים שלמים ובודקת אם הוא "עץ מעולה". אם כן – הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

(3 נק') ג. נתונה טענה: ערך <u>הכי קטן</u> ב"**עץ מעולה**" נמצא בעלה <u>הכי שמאלי</u>. האם הטענה נכונה? **הסבירו את תשובתכם.**

(3 נק') ד. מהי הסיבוכיות של הפעולות שכתבתם? **הסבירו את תשובתכם.**

שאלה 8

נתונות המחלקות האלה:

```
public abstract class A {
  public int x;

  public A() {
   this.x = 1;
  }
  public abstract int F(int v);

  public virtual void F(A a) {
    x = a.x;
  }

  public override
          string ToString(){
    return "x =" + this.x;
  }
} //end of class A
```

```
public class B : A {
 public B():base() {
}
 public B(int val):base(){
   this.x = F(val);
 }
 public override int F(int val){
   return this.x + val;
 }
 public virtual void F(B b){
   this.x = this.x * b.x;
 }
 public override
      string ToString(){
 return "B: " + base.ToString();
 }
} //end of class B
```

```
public class D : B {
  public D(int val):base(){
        this.x = val - this.x;
  }

  public override void F(A a){
    this.x = this.x + a.x + 1;
  }
  public override void F(B b){
    this.x = this.x * b.x;
  }
  public void F(D d)  {
    this.x = d.x - 1;
  }
 public override string ToString(){return "D: " + base.ToString();}

} //end of class D
```

```
public class Driver {
  public static void Main (string [] args){
   A a;
   B b;
   int m;
//    ************
 }
}
```

בכל אחד מסעיפים הבאים השורה (*****) תוחלף לקטע קוד רלוונטי.

(4 נק׳)א. מהו הפלט של קטע הקוד שלפניכם:

```
m = 11;
a = new B(m);
Console.WriteLine(a);
a = new D(m);
Console.WriteLine(a);
```

(4 נק׳)ב. מהו הפלט של קטע הקוד שלפניכם:

```
b = new B(3);
m = b.F(4);
Console.WriteLine(m);
m = b.F(m);
Console.WriteLine(m);
```

(3 נק׳)ג. מהו הפלט של קטע הקוד שלפניכם:

```
a = new B(5);
m = a.F(a.F(2));
Console.WriteLine(m);
```

(3 נק׳)ד. מהו הפלט של קטע הקוד שלפניכם:

```
a = new D(10);
b = new B(20);
a.F(b);
Console.WriteLine(a);
```

(4 נק׳)ה. אחרי ביצוע קטע הקוד הבא, הערך של התכונה `x` של העצם `d` יהיה 5.

מהו הערך של המשתנה `m`?

```
D d = new D(m);
b = new B();
((A)d).F(b);
```

<u>שאלה 9</u>

נתונות שתי הפעולות IsExist1 ו- IsExist2

```
public static bool IsExist1(Stack<int> st, int x){
    while(!st.IsEmpty()){
        if(st.Pop()==x) return true;
    }
    return false;
}
public static bool IsExist2(Stack<int> st, int x){
    Stack<int> temp = new Stack<int>();
    bool f = false;
    while(!st.IsEmpty()){
        if(st.Top()==x) f = true;
        temp.Push(st.Pop());
    }
    while(!temp.IsEmpty()) {
        st.Push(temp.Pop());
    }
    return f;
}
```

(3 נק׳) א. הסבירו מה מבצעת כל אחת מהפעלות, מה הבדל ביניהן ומהי הסיבוכיות של כל אחת מהן.

(4 נק׳) ב. כתבו פעולה המקבלת שתי מחסניות של מספרים שלמים st1 ו-st2. הפעולה תבדוק אם כל האיברים של המחסנית st1 נמצאים במחסנית st2. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור זוג המחסניות שלפניכם הפעולה תחזיר true:

st1	2	3	10	7

st2	3	14	2	2	3	7	10	2	10

(4 נק׳) ג. כתבו פעולה המקבלת שתי מחסניות של מספרים שלמים st1 ו-st2. הפעולה תבדוק אם כל האיברים של המחסנית st1 נמצאים במחסנית st2 **באותו הסדר**. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור זוג המחסניות מסעיף ב׳ הפעולה תחזיר false. אבל אם המחסנית st2 תהיה כך:

st2	3	14	2	2	3	7	10	2	*7*

הפעולה תחזיר ערך true.

(4 נק׳) ד. כתבו פעולה המקבלת שתי מחסניות של מספרים שלמים st1 ו-st2. הפעולה תבדוק אם המחסנית st1 היא תת-מחסנית של st2 (כלומר, אם כל האיברים של המחסנית st1 נמצאים במחסנית st2 **ברצף ובאותו הסדר**. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור זוג המחסניות מסעיף ב׳ או ג׳ הפעולה תחזיר false. אבל אם המחסנית st2 תהיה כך:

st2	3	14	2	2	3	*10*	*7*	2	7

הפעולה תחזיר ערך true.

(3 נק׳) ה. מהי הסיבוכיות של כל אחת מהפעולות שכתבתם? הסבירו את תשובתכם.

מבחן ב-JAVA

הנחיות כלליות לנבחנים:

בנספחי הבחינה מובאים ממשקים (interface) הכוללים פונקציות ומחלקות עזר, יש להקפיד ולהשתמש בהן בשאלות הרלוונטיות בבחינה על-מנת להימנע מאיבוד ניקוד.

מבחן ב-JAVA

חלק א'

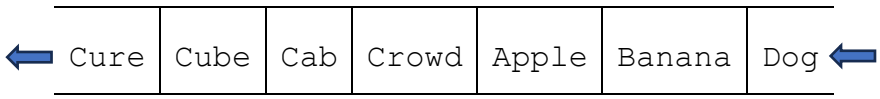
ענו על שתיים מבין השאלות 1–3 (ערך כל שאלה – 16 נקודות).

שאלה 1

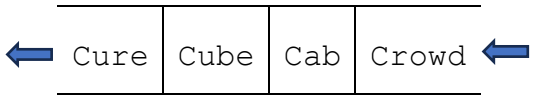
הגדרה:

תור מחרוזות "תקין" הוא תור שבו כל המחרוזות המתחילות באותה האות נמצאות רק בתחילת התור.

לדוגמה, התור שלפניכם הוא "תור תקין":



גם התור הזה הוא "תור תקין":



אבל התור הזה אינו "תור תקין":

(3 נק') א. כתבו פעולה המקבלת תור של מחרוזות "**תקין**" `q` ומחרוזת `st`. הפעולה תוסיף את המחרוזת לתור כך שהוא יישאר תקין.

כותרת הפעולה:

```
public static void addToProperQueue(Queue<String> q, String s)
```

(5 נק') ב. כתבו פעולה המקבלת תור של מחרוזות. הפעולה תבדוק אם תור הוא "תור תקין". אם כן – הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

כותרת הפעולה:

```
public static boolean isProperQueue(Queue<String> q)
```

(5 נק') ג. כתבו פעולה המקבלת תור של מחרוזות. הפעולה תבדוק אם התור הוא "תור תקין". אם כן – הפעולה לא תבצע דבר, ואם לא – הפעולה תשנה את סדר המחרוזות בתור כך שהוא יהיה "**תקין**".

כותרת הפעולה:

```
public static void fixIt(Queue<String> q)
```

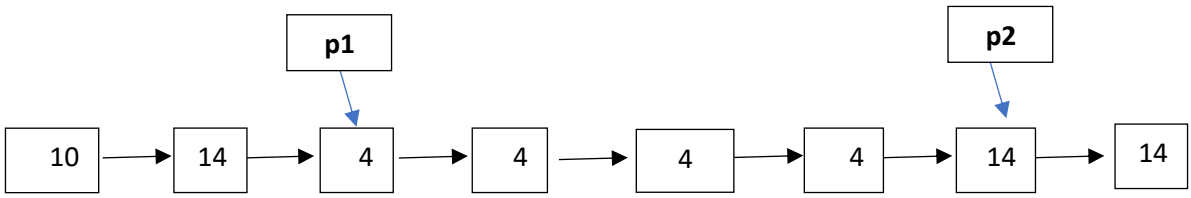
(3 נק') ד. מהי סיבוכיות הפעולות שכתבתם בסעיפים א'–ג'? **הסבירו את תשובתכם.**

<u>שאלה 2</u>

נגדיר "רצף K" כרצף של K חוליות של מספרים שלמים שהערך של כל אחת מהן הוא K.

(6 נק') א. כתבו פעולה המקבלת שתי הפניות p1 ו-p2 לחוליות שונות בשרשרת חוליות של מספרים שלמים. הפעולה תבדוק אם החוליות בין p1 (כולל) ל-p2 (לא כולל) מהוות את "רצף K". אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור השרשרת שלפניכם הפעולה תחזיר ערך true:



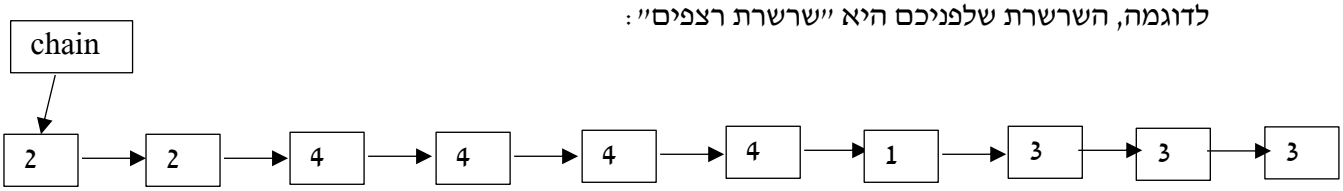
כותרת הפעולה:

```
public static boolean isSequenceK(Node<Integer> p1,
                                  Node<Integer> p2)
```

(8 נק') ב. שרשרת חוליות נקראת "שרשרת רצפים" אם היא מורכבת מכמה "רצפי K".

כתבו פעולה המקבלת הפניה לחוליה הראשונה של שרשרת חוליות של מספרים שלמים. הפעולה תבדוק אם השרשרת היא "שרשרת רצפים". אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, השרשרת שלפניכם היא "שרשרת רצפים":



כותרת הפעולה:

```
public static boolean isSequenceList(Node<Integer> chain)
```

(2 נק') ג. מהי סיבוכיות הפעולות שכתבתם בסעיפים א'–ב'? **הסבירו את תשובתכם.**

שאלה 3

בשאלה זו שלושה סעיפים. אין קשר בין הסעיפים.

(4 נק׳) א. נתונה פעולה ראשית במחלקה `Test`.

1. שרטטו את **עצי הירושה עבור שתי** אפשרויות היררכיה בין המחלקות כך שהפעולה הראשית תהיה תקינה והתוכנית תרוץ ללא שגיאות.

```
public class Test {
    public static void main(String[] args){
        S s1 = new R();
        P p1 = (P)(new R());
        S s2 = new P();
        P p2 = new Q();
//      (*****)
    }
}
```

2. רוצים להוסיף בסוף התוכנית את שורת הקוד שלפניכם:

```
Q q1 =(Q) p1;
```

האם התוכנית הראשית תהיה תקינה עבור שתי האפשרויות שציינתם? אם לא – מהו סוג השגיאה שתתרחש?

(6 נק׳) ב. לפניכם המחלקות `First`, `Second` ו-`Program`. במחלקות `First` ו-`Second` **חסר מימוש בנאים ופעולות** `toString`.

```
public class First {
 private static int c = 0;
 protected int num;
 public First(int num){…}
 public String toString(){…}
}
```

```
public class Second extends First{
 private char ch;
 public Second(int num, char ch){…}
 public String toString(){…}
}
```

השלימו את הפעולות במחלקות `First` ו-`Second` כך שאחרי הרצת הפעולה הראשית של המחלקה `Program` יתקבל הפלט הזה:

```
public class Program{
 public static void main(String[] args) {
  First f1 = new First(10);
  System.out.println("f1 is "+f1);
  First f2 = new Second(3,'A');
  System.out.println("f2 is "+f2);
  First f3 = new Second(4,'Z');
  System.out.println("f3 is "+f3);
  First f4 = new First(7);
  System.out.println("f4 is "+f4);
 }
}
```

הפלט שהתקבל אחרי הרצת התוכנית:

```
f1 is 1-10
f2 is 2-6-A
f3 is 3-12-Z
f4 is 4-28
```

(6 נק׳) ג. לפניכם קוד המחלקה `Two`. המחלקה יורשת מהמחלקה `One`.

```
public class Two extends One
{
    private int z;
    public Two()
    {
        this.z = 11;
    }

    public Two(String s1, int z)
    {
        super(s1);
        this.z = z;
    }

    public double p1(int a)
    {
        return a * super.p1(a);
    }

    public int p2()
    {
        return One.w + this.m();
    }

    public void print()
    {
        One.d();
        System.out.println(s+"SON");
    }
}
```

כתבו במחלקה `One` את התכונות וה**כותרות** של הפעולות הנדרשות.

יש לציין הרשאת הגישה (`public`, `protected`, `private`) לכל תכונה ולכל פעולה.

אין צורך לממש את הפעולות.

אין לשנות את הקוד של המחלקה `Two`.

חלק ב׳

ענו על <u>שתיים</u> מבין השאלות 4–6 (ערך כל שאלה – 16 נקודות).

<u>שאלה 4</u>

החברה ״סוד זה הצלחה״ החליטה שכל הישיבות והכנסים שלה יתקיימו באופן וירטואלי דרך מערכת מפגשים וירטואליים Boom שפיתחו מתכנתים שלה. במפגשי Boom ניתן לחלק משתתפים בין ״חדרים וירטואליים״. לשם ניהול המפגשים הווירטואליים פותחו מחלקות Meeting, VRoom, Participant.

- כל משתתף (Participant) מאופיין על ידי שם (name) ומספר היחידה שבה הוא עובד (dep, מספר שלם בין 1 ל-10).
- כל חדר (VRoom) מאופיין על ידי מספר חדר (num, מספר שלם וחיובי) ותור משתתפים (parts).
- למחלקה Meeting תכונה אחת – רשימת חדרים וירטואליים (rooms - שרשרת חוליות מסוג VRoom).

```
public class User {
 private String name;
 private int dep;
 public User(String n, int d){
  this.name = n;
  this.dep = d;
 }
 } //אפשר להניח שהוגדרו הפעולות set/get
```

```
public class Meeting {
 private Node<VRoom> rooms;
 public Meeting() {
  this.rooms = null;
 }
}
```

```
public class Vroom {
 private int num;
 Queue<Participant> parts;
 public VRoom(int n){… }
}
```

(2 נק׳) א. כתבו במחלקה VRoom פעולה בונה המקבלת מספר חדר ובונה חדר ריק ממשתתפים.

(2 נק׳) ב. כתבו במחלקה VRoom פעולה add המקבלת משתתף ומוסיפה אותו לחדר.

(6 נק׳) ג. לפני כל ישיבה, הנהלת החברה מחליטה על מספר המשתתפים שיהיו בחדר וירטואלי.
כתבו במחלקה Meeting פעולה המקבלת מספר שלם וחיובי number. הפעולה תקלוט את פרטי המשתתפים (שם ומספר יחידה). הפעולה תיצור חדר שמספרו אחד ואליו יצטרפו number משתתפים ראשונים, לאחריו חדר מספר שתיים ואליו יצטרפו number משתתפים הבאים וכך הלאה. הקלט יסתיים כאשר ייקלט 0 כמספר המחלקה.
כותרת הפעולה:

```
public void divide (int number)
```

(6 נק׳) ד. כאשר מטרת הישיבה היא סיעור מוחות (brainstorming), חשוב שבכל חדר יהיו נציגים מכל עשר יחידות החברה.

כתבו במחלקה Meeting פעולה אשר מדפיסה עבור כל חדר את מספרי יחידות החברה שנציגיהן לא נמצאים בחדר.
כותרת הפעולה:

```
public void printMissingUnits()
```

שאלה 5

(2 נק׳) א. כתבו פעולה **getLast** המקבלת הפניה לחוליה הראשונה של שרשרת של מספרים שלמים.
הפעולה תחזיר הפנייה לחוליה האחרונה של השרשרת.

נתונה הפעולה **what** המקבלת שני מספרים שלמים **k<n**. הפעולה משתמשת בפעולה getLast.

```
public static Node<Integer> what (int n, int k){
     if (n == k)
         return new Node<Integer>(n);
     else
     {
         if (k % 2 == 0)
             return new Node<Integer>(k, what(n, k+1));
         else
         {
             Node<Integer> p = new Node<Integer>(k);
             Node<Integer> chain = what (n, k+1);
             Node<Integer> last = getLast (chain);
             last.setNext (p);
             return chain;
          }
     }
 }
```

(6 נק׳) ב. עקבו אחרי זימון הפעולה what(10,5) ורשמו מה תחזיר הפעולה.

יש להראות את תוכן השרשרת בחזרה מכל קריאה הרקורסיבית.

נתונה הפעולה **secret**:

```
public static Node<Integer> secret (int n, int k) {
     if (n == k)
         return new Node<Integer>(n);
     else
     {
         if (n % 2 == 0)
             return new Node<Integer>(n, secret(n-1, k));
         else
         {
             Node<Integer> chain = secret (n-1, k);
             getLast (chain).setNext (new Node<Integer>(n);
             return chain;
         }
     }
 }
```

(3 נק׳) ג. מה תהיה תוצאת הזימון secret(6, 2)?

(2 נק׳) ד. האם קיימים שני מספרים שלמים וחיוביים k<n כך שתוצאות הזימונים secret(n, k) ו-what(n, k) יהיו זהות? אם כן – תנו דוגמה לזוג מספרים שכזה, ואם לא – הסבירו למה הדבר אינו אפשרי.

(3 נק׳) ה. כתבו את הפעולה what בצורה הלא רקורסיבית.

שאלה 6

(4 נק') א. כתבו פעולה `sumLeaves` שמקבלת עץ בינארי של מספרים שלמים ומחזירה את סכום הערכים של **כל העלים** בעץ.

נתונה המחלקה `NodeInfo` שמכילה שתי תכונות: מספר שלם `num`, ומחרוזת `info`.
במחלקה הוגדרה פעולה בונה:

```
public NodeInfo(int num, String info)
```

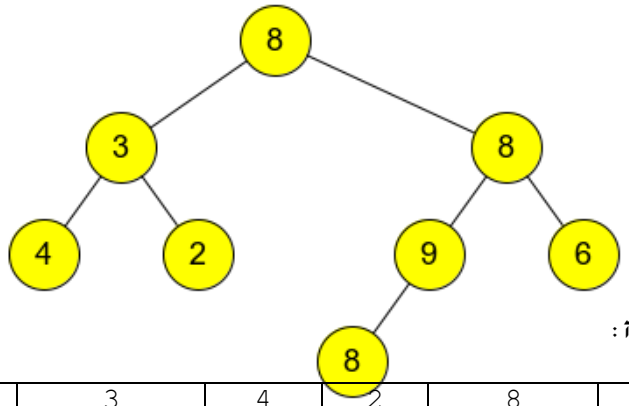
במחלקה הוגדרו פעולות `get` ו-`set` במחלקה `NodeInfo`.

(6 נק') ב. כתבו פעולה `createQueue` המקבלת עץ בינארי של מספרים שלמים ומחזירה תור מטיפוס `NodeInfo` שבו:

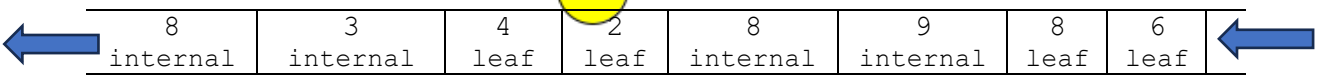
- ערך ה-`num` בכל איבר זהה לערך הצומת בעץ המתקבל כפרמטר.
- ערך ה-`info` מכיל את המחרוזת "*leaf*" אם הצומת הוא עלה, ואת המחרוזת "*internal*" אם הצומת הוא צומת פנימי.

סדר האיברים בתור לא חשוב!

לדוגמה, עבור העץ שלפניכם:



הפעולה תחזיר את תור הזה:



כותרת הפעולה:

```
public static Queue<NodeInfo> createQueue (BinNode<Integer> bt)
```

(6 נק') ג. כתבו פעולה `maxLeaveValue` המקבלת עץ בינארי של מספרים שלמים ומחזירה את **הערך הגדול ביותר מבין העלים של העץ.**

חלק ג'

ענו על <u>שתיים</u> מבין השאלות 7–9 (ערך שאלה – 18 נקודות).

<u>שאלה 7</u>

(6 נק') א. כתבו פעולה המקבלת שני תורים של מספרים שלמים `q1` ו-`q2`. הפעולה תבדוק אם כל הערכים של תור `q1` קטנים מכל הערכים של `q2`. אם כן, הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

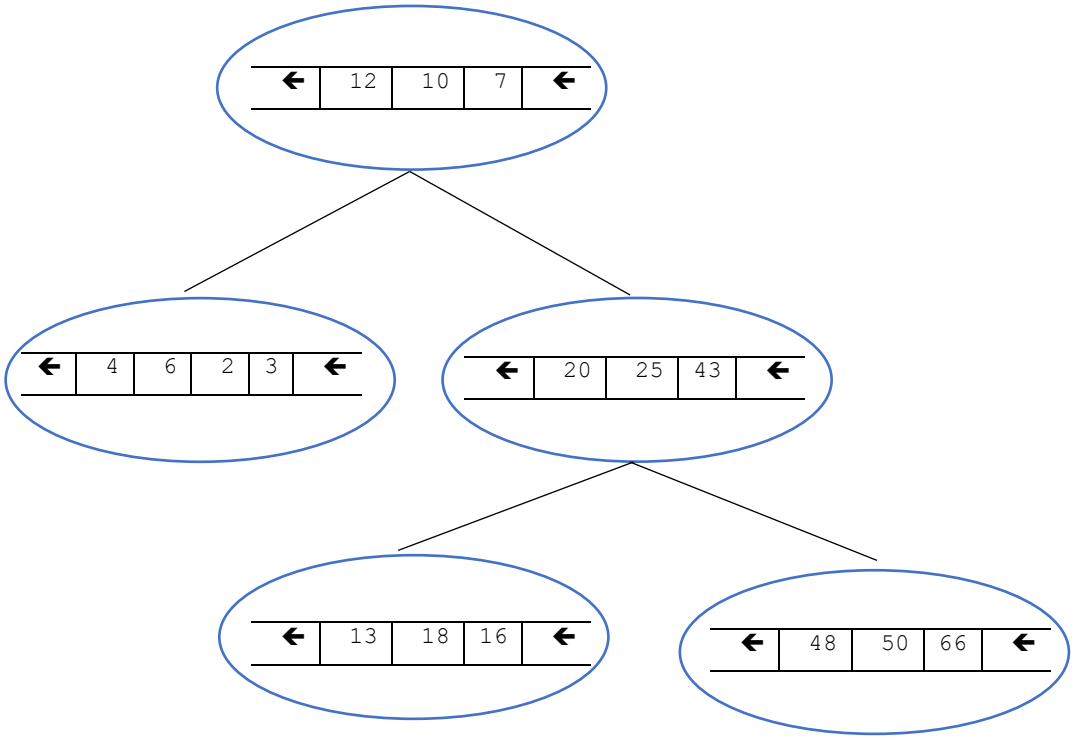
כותרת הפעולה:

```
public static boolean isSmaller(Queue<Integer>q1,
                                Queue<Integer> q2)
```

(6 נק') ב. נתון עץ שבו כל צומת מכיל תור של מספרים שלמים. עץ נקרא "עץ מעולה" אם הוא עונה על שני התנאים האלה:

- לכל צומת שאינו עלה יש שני בנים.
- כל הערכים הנמצאים בצומת האב גדולים מכל הערכים הנמצאים בבן השמאלי וקטנים מכל הערכים הנמצאים בבן הימני.

לדוגמה, העץ שלפניכם הוא "עץ מעולה":



כתבו פעולה המקבלת עץ תורים של מספרים שלמים ובודקת אם הוא "עץ מעולה". אם כן – הפעולה תחזיר ערך `true`, ואם לא – הפעולה תחזיר ערך `false`.

(3 נק') ג. נתונה טענה: ערך <u>הכי קטן</u> ב"**עץ מעולה**" נמצא בעלה <u>הכי שמאלי</u>. האם הטענה נכונה? **הסבירו את תשובתכם.**

(3 נק') ד. מהי הסיבוכיות של הפעולות שכתבתם? **הסבירו את תשובתכם.**

שאלה 8

נתונות המחלקות האלה:

```
public abstract class A
{
  protected int x;

  public A() {
   this.x = 1;
  }
  public abstract int f(int v);
  public void f(A a) {
    x = a.x;
  }
  public String toString(){
    return "x =" + this.x;
  }
} //end of class A
```

```
public class B extends A {
 public B() {
   super();
 }
 public B(int val) {
   this.x = f(val);
 }
 public int f(int val) {
   return this.x + val;
 }
 public void f(B b) {
   this.x = this.x * b.x;
 }
 public String toString(){
  return "B: " +
          super.toString();
 }
} //end of class B
```

```
public class D extends B {
  public D(int val)  {
        this.x = val - this.x;
  }

  public void f(A a)  {
    this.x = this.x + a.x + 1;
  }

  public void f(B b)  {
    this.x = this.x * b.x;
  }

  public void f(D d)  {
    this.x = d.x - 1;
  }
  public String toString(){
    return "D: " +
          super.toString();
 }

} //end of class D
```

```
public class Driver
{
  public static void main
          (String [] args){
   A a;
   B b;
   int m;
//    ************
 }
}
```

בכל אחד מסעיפים הבאים השורה (*****) תוחלף לקטע קוד רלוונטי.

(4 נק׳)א. מהו הפלט של קטע הקוד שלפניכם:

```
m = 11;
a = new B(m);
System.out.println(a);
a = new D(m);
System.out.println(a);
```

(4 נק׳)ב. מהו הפלט של קטע הקוד שלפניכם:

```
b = new B(3);
m = b.f(4);
System.out.println(m);
m = b.f(m);
System.out.println(m);
```

(3 נק׳)ג. מהו הפלט של קטע הקוד שלפניכם:

```
a = new B(5);
int m = a.f(a.f(2));
System.out.println(m);
```

(3 נק׳)ד. מהו הפלט של קטע הקוד שלפניכם:

```
a = new D(10);
b = new B(20);
a.f(b);
System.out.println(a);
```

(4 נק׳)ה. אחרי ביצוע קטע הקוד הבא, הערך של התכונה `x` של העצם `d` יהיה 5.

מהו הערך של המשתנה `m`?

```
D d = new D(m);
b = new B();
((A)d).f(b);
```

<u>שאלה 9</u>

נתונות שתי הפעולות isExist1 ו- isExist2

```
public static boolean isExist1(Stack<Integer> st, int x){
    while(!st.isEmpty()){
        if(st.pop()==x) return true;
    }
    return false;
}
public static boolean isExist2(Stack<Integer> st, int x){
    Stack<Integer> temp = new Stack<Integer>();
    boolean f = false;
    while(!st.isEmpty()){
        if(st.top()==x) f = true;
        temp.push(st.pop());
    }
    while(!temp.isEmpty()) {
        st.push(temp.pop());
    }
    return f;
}
```

(3 נק׳) א. הסבירו מה מבצעת כל אחת מהפעלות, מה הבדל ביניהן ומהי הסיבוכיות של כל אחת מהן.

(4 נק׳) ב. כתבו פעולה המקבלת שתי מחסניות של מספרים שלמים st1 ו-st2. הפעולה תבדוק אם כל האיברים של המחסנית st1 נמצאים במחסנית st2. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור זוג המחסניות שלפניכם הפעולה תחזיר true:

st1	2	3	10	7

st2	3	14	2	2	3	7	10	2	10

(4 נק׳) ג. כתבו פעולה המקבלת שתי מחסניות של מספרים שלמים st1 ו-st2. הפעולה תבדוק אם כל האיברים של המחסנית st1 נמצאים במחסנית st2 **באותו הסדר**. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור זוג המחסניות מסעיף ב׳ הפעולה תחזיר false. אבל אם המחסנית st2 תהיה כך:

st2	3	14	2	2	3	7	10	2	*7*

הפעולה תחזיר ערך true.

(4 נק׳) ד. כתבו פעולה המקבלת שתי מחסניות של מספרים שלמים st1 ו-st2. הפעולה תבדוק אם המחסנית st1 היא תת-מחסנית של st2 (כלומר, אם כל האיברים של המחסנית st1 נמצאים במחסנית st2 **ברצף ובאותו הסדר**. אם כן, הפעולה תחזיר ערך true, ואם לא – הפעולה תחזיר ערך false.

לדוגמה, עבור זוג המחסניות מסעיף ב׳ או ג׳ הפעולה תחזיר false. אבל אם המחסנית st2 תהיה כך:

st2	3	14	2	2	3	*10*	*7*	2	7

הפעולה תחזיר ערך true.

(3 נק׳) ה. מהי הסיבוכיות של כל אחת מהפעולות שכתבתם? הסבירו את תשובתכם

נספח ממשקים מבנה הנתונים בתוכנית הלימודים-JAVA

ממשק החוליה 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<T>**()
הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אחרת	**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): $[x_1, x_2, ..., x_n]$	**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<T>**()
הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ו"שקר" אחרת	**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): $[x_1, x_2, ..., x_n]$	**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.

ממשק המחלקה הגנרית - 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 חוליה משמאל	**boolean** hasLeft()
הפעולה מחזירה **true** אם יש חוליה מימין	**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 ממשקים מבנה הנתונים בתוכנית הלימודים - C#

ממשק החוליה 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	**string** ToString()

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

ממשק המחלקה מחסנית STACK<T>

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

הפעולה בונה מחסנית ריקה	Stack<T>()
הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אחרת	**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 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): $[x_1, x_2, ..., x_n]$	**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 תור QUEUE<T>

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

Queue<T>()	הפעולה בונה תור ריק
bool IsEmpty()	הפעולה מחזירה"אמת" אם התור הנוכחי ריק, ו"שקר" אחרת
void Insert (**T** x)	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): $[x_1, x_2, ..., x_n]$

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 חוליה בינרית BINNODE<T>

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

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

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

97105 מבני נתונים ותכנות מונחה עצמיים– מועד א' קיץ 2025

שאלה	סעיף	תת-סעיף	ניקוד	הערות
1	א	-	3	
	ב	-	5	*חריגה – להוריד 1 נק'*
	ג	-	5	*איבד איברים – להוריד 2 נק'*
	ד	-	3	*בלי הסבר – לא לתת נקודות*
2	א	-	6	*חריגה – להוריד 2 נקודות* *לולאה אין סופית – להוריד 2 נק'* *הכניס p2 לבדיקה – להוריד 1 נק'*
	ב	-	8	*לא השתמש בפעולה של סעיף א' ולא עדכן מונה לכל רצף – לוריד 2 נק'* *החזיר true אמצע הלולאה – להוריד 2 נק'*
	ג		2	*בלי הסבר – לא לתת נקודות*
3	א	-	4	1. 1.5 נק' עבר כל עץ 2. 1 נק' – *לא ציין סוג שגיאה – לא לתת נקודות*
	ב	-	6	• כל בנאי – 1 נק' • כל פעולה TOSTRING – 2 נק'
	ג		6	• תכונות – 2 נק' • פעולות – 4 נק' *לא ציין Static – להוריד 1 נק' לכל פעולה*
4	א	-	2	•
	ב	-	2	•
	ג	-	6	• לולאה קליטה – 2 נק' • יצירת משתתף – 1 נק' • יצרת חדר – 1 נק' • הוספת חדר ל-ROOMS– 2 נק'
	ד	-	6	• סריקה ROOMS – 2 נק' • בדיקה עבור כל חדר - 3 נק' • הדפסה – 1 נק'
5	א	-	2	
	ב	-	6	***בלי מעקב לא לתת נקודות!***
	ג	-	3	*אין צורך במעקב, תשובה לא נכונה – לא לתת נקודות*
	ד	-	2	*בלי הסבר – לא לתת נקודות*
	ה	-	3	
6	א	-	4	***כתב פעולה leaf(tree) לא תקינה – להוריד 1 נק' פעם אחת!*** *כתב פעולה המחזירה סכום צמתים בעץ – להוריד 3 נק'*
	ב	-	6	*אתחל תור בכל זימון רקורסיבי – להוריד 2 נק'* *לא יצר עצם – להוריד 1 נק'* *פנה לתת עץ בלי לבדוק שקיים – להוריד 1 נק'*
	ג	-	6	*הניח שערכי העץ כולם חיוביים – להוריד 1 נק'*
7	א	-	6	*חריגה – להוריד 2 נקודות* *לולאה אין סופית – להוריד 2 נק'* *החזיר TRUE באמצע הלולאה – להוריד 2 נק'*
	ב	-	6	• בדיקה תנאים – 4 נק' • זימון רקורסיבי –2 נק'
	ג		3	*בלי הסבר – לא לתת נקודות.* ***יש להתייחס לפתרון!***
	ג	-	3	*בלי הסבר – לא לתת נקודות.* ***יש להתייחס לפתרון!***

8	א	-	4	
	ב	-	4	
	ג	-	3	
	ד	-	3	
	ה	-	4	
9	א	-	3	*חריגה – להוריד 1 נקודות* *לולאה אין סופית – להוריד 1 נק'* *בלי הסבר – לא לתת נקודות*
	ב	-	4	
	ג	-	4	
	ד	-	4	
	ה	-	3	