GuyAI
מבית GOOL בגרויות

סוג הבחינה: בחינת גמר ממלכתית
מועד הבחינה: קיץ 2023 מועד ב'
סמל השאלון: 97105
קובץ: תשובות סופיות

# מבנה נתונים ותכנות מונחה עצמים

## הנדסת תוכנה · הנדסאים וטכנאים

### הוראות

**א.** משך הבחינה: ארבע שעות וחצי.

**ב.** מבנה השאלון ומפתח ההערכה: **בשאלון זה שני מבחנים, עליכם לענות על מבחן אחד בלבד בהתאם למוסד הלימודים:** מבחן ב- Java (עמוד 2) מבחן ב- C# (עמוד 15) 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 בשימוש: 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 מיוחדות: 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.

השאלות בשאלון זה מנוסחות בלשון רבים, אף על פי כן על כל תלמידה וכל תלמיד להשיב עליהן באופן אישי.

**בהצלחה!**

# תשובות סופיות

### שאלה 1

**א.** עוברים על כל המחלקים האפשריים $d$ מ-2 עד $\min(num1, num2)$; אם $d$ מחלק את שני המספרים מחזירים `false`, ואם לא נמצא מחלק כזה מחזירים `true`.

**ב.** מרוקנים את `q` לשני תורי עזר: הזרים ל-`num` (לפי `strangers` מסעיף א') והלא זרים; אחר כך מחזירים ל-`q` קודם את כל תור הזרים ואחריו את תור הלא זרים.

**ג.** סעיף א': $O(\min(num1, num2))$. סעיף ב': $O(n \cdot num)$, כאשר $n$ מספר האיברים בתור (אם מתייחסים לבדיקת זרות כפעולה בגודל חסום - $O(n)$).

### שאלה 2

**א.** שומרים את זוגיות הערך הראשון ועוברים על השרשרת עם `prev` ו-`cur`; כל חוליה שזוגיותה שונה מנותקת מהשרשרת המקורית (`prev.setNext(cur.getNext())`) ומחוברת לסוף השרשרת החדשה. מחזירים את ראש השרשרת החדשה (או `null`).

**ב.** $O(n)$ - מעבר יחיד על השרשרת, ובכל חוליה מספר קבוע של פעולות.

### שאלה 3

**א.** ב-`First`: בונה ריקה (`num=10`) ובונה עם `num`. ב-`Second`: בונה ריקה (`num=10, x=1.1,`) `f=null`); בונה `(int num, First other)` עותק חדש של `other`; בונה `(int a, double b, First f)` מפנה לעצם שהתקבל. `num=(int)(a+b)=5`, `x=num+0.5=5.5`, `f` - `num`, `x=num+0.5`, `f` -

**ב.**
(1) לא תקין - שגיאת קומפילציה (הידור).
(2) לא תקין - שגיאת קומפילציה (הידור).
(3) תקין.
(4) תקין.
(5) לא תקין - שגיאת זמן ריצה.
(6) לא תקין - שגיאת זמן ריצה (בשורה השלישית).
(7) תקין.
(8) לא תקין - שגיאת קומפילציה (הידור).

**שאלה 4**

(1) לא תקין – שגיאת קומפילציה: אי אפשר ליצור עצם מממשק.

(2) תקין. הפלט: `ctor One`, `ctor1 Three`, `Three: doIt2()`, `One:DoIt1()`, `One: a=5 ctor One`, `ctor1 Three`, `Three: doIt2()`, `One:DoIt1()`, `One: a=5 Three: c=10`, `One: a=5 Three: c=10`.

(3) לא תקין – שגיאת קומפילציה: אי אפשר להציב עצם `One` במשתנה `Three`.

(4) לא תקין – שגיאת זמן ריצה בהמרה `x4(Three)`; לפני השגיאה מודפס `ctor One`, `ctor Two`.

(5) תקין. הפלט: `ctor One`, `ctor1 Three`, `Three: doIt1()`, `One:DoIt1()`, `One: a=5 ctor One`, `ctor1 Three`, `Three: doIt1()`, `One:DoIt1()`, `One: a=5 Three: c=5`.

**שאלה 5**

**א.** מוציאים איברים מראש המחסנית למחסנית עזר וסופרים; בהופעה הראשונה של `num` שומרים את המונה ועוצרים. מחזירים את כל האיברים מהעזר ל-`stk` ומחזירים את המיקום (או 1-). בדוגמה: 2.

**ב.** מוציאים את כל המחסנית למחסנית עזר תוך ספירה, ובכל הופעה של `num` מעדכנים `place = count`; כך נשאר המיקום של ההופעה האחרונה. מחזירים את האיברים ל-`stk` ומחזירים `place`. בדוגמה: 10.

**ג.** מוציאים את האיברים אחד-אחד למחסנית עזר; לכל איבר `x` שהוצא מזמנים `firstPlace(stk, x)` על החלק שנשאר. אם בן הזוג נמצא במקום `place`, המרחק הוא `place - 1`; שומרים את המינימום. בסוף מחזירים את האיברים ל-`stk`. בדוגמה: 1.

**ד.** א' – $O(n)$, ב' – $O(n)$, ג' – $O(n^2)$, כאשר $n$ מספר האיברים במחסנית.

**שאלה 6**

**א.** מחלקה מופשטת `Shoe` (producer, size, color, price); `KidsShoe` יורשת ממנה (orth); מחלקה מופשטת `AdultShoe` יורשת מ-`Shoe` (year); `WomenShoe` (heel)- ו-`MenShoe` (work) יורשות מ-`AdultShoe`.

**ב.** כל מחלקה מגדירה רק את התכונות החדשות שלה; הבונה של כל מחלקה יורשת מזמנת את בונה מחלקת הבסיס. המחיר הבסיסי נקבע בבונה: 100 ב-`KidsShoe`, 200 ב-`AdultShoe`.

**ג.** `Storage` מכילה מערך `Shoe[] shoes` (מטיפוס מחלקת הבסיס, כך שאפשר לשמור בו כל סוג נעל) ומונה `count`; ב-UML – קשר הכלה מ-`Storage` אל `Shoe`.

**ד.** `Shoe.calcPrice()` – מחיר בסיסי או 300 למידה מעל 40; `KidsShoe` דורסת ומוסיפה 25 לאורתופדית; `AdultShoe` דורסת: מוסיפה `extra()` (מופשטת) ומכפילה ב-1.5 לשנת 2023; `WomenShoe.extra()` – 30/60 לפי העקב, `MenShoe.extra()` – 50 לנעל עבודה.

**ה.** עוברים על `shoes[0..count-1]`; סופרים כל איבר שהוא `KidsShoe` (בדיקת `is` / `instanceof`) ושה-`isOrth()` שלו `true`.

### שאלה 7

**א.** כן. למשל שורש 5 שבנו השמאלי 5 (ואין בן ימני): בכל צומת אין ערך משותף בין תת-העץ השמאלי לימני, אף שהערך 5 מופיע פעמיים.

**ב.** לא. בעץ חיפוש בינרי, בכל צומת כל ערכי תת-העץ השמאלי קטנים מערך הצומת וכל ערכי תת-העץ הימני גדולים ממנו, ולכן אין ביניהם ערך משותף - כל עץ חיפוש בינרי הוא "עץ מיוחד".

**ג.** רקורסיה: עץ ריק מיוחד; אחרת בודקים שאף ערך של תת-העץ השמאלי אינו מופיע בתת-העץ הימני (`noCommon`, בעזרת חיפוש `exists`), ושני תתי-העצים מיוחדים בעצמם.

**ד.** $O(n^2)$, כאשר $n$ מספר הצמתים בעץ.

### שאלה 8

**א.** `B` יורשת מ-`A` ו-`C` יורשת מ-`B` (חצי ירושה C→B→A); בנוסף `C` מכילה עצם `A` (התכונה `myA`) - קו הכלה עם מעוין מלא בצד של `C`.

**ב.** נוצרו 7 עצמים: arr[0] - A(x=13); arr[1] - B(x=5, y=10); arr[2] - C(x=2, y=7, myA→A(x=9)); arr[3] - C(x=3, y=3, myA→A(x=6)); arr[4] - B(x=0, y=5). המונים: countA=7, countB=4, countC=2.

**ג.** `A objects: 7`, `B objects: 4`, `C objects: 2`.

**ד.** `A:13`, `B:A:5:10`, `C:B:A:2:7 [A:9]`, `C:B:A:3:3 [A:6]`, `B:A:0:5`.

**ה.** `20`, `13`, `15`, `5` (עבור k=1, 2, 3, 4).

### שאלה 9

**א.** התור יהיה: 3, 7, 2, 4, 9-, 1- (ראש התור 3).

**ב.** (1) לא נכונה. דוגמה נגדית: התור 1, 2 (ראש התור משמאל). שני הערכים חיוביים, חוזרים לסוף התור בסדרם המקורי, ואין שליליים - התור נשאר 1, 2, למרות שיש בו שני ערכים.

(2) לא נכונה. דוגמה נגדית: התור 5- (ערך שלילי אחד). אין חיוביים, והשלילי היחיד "מתהפך" לעצמו - התור נשאר 5-. גם התור 3, 1- נשאר ללא שינוי.

(3) לא נכונה. דוגמה נגדית: התור 1, 2 - כולו חיובי, ונשאר ללא שינוי.

(4) לא נכונה. דוגמה נגדית: התור 3, 1- (ראש התור 3). החיובי 3 נשאר בתחילת התור, והשלילי היחיד 1- נשאר אחריו - התור נשאר 3, 1-. (גם 4, 1-, 2-, 1- נשאר ללא שינוי.)

**ג.** מסדרת את התור כך שכל המספרים החיוביים יהיו בתחילתו, בסדרם המקורי, ואחריהם כל המספרים השליליים בסדר הפוך לסדרם המקורי.

**ד.** מוסיפים 0; כל עוד ראש התור אינו 0 - מוציאים איבר: חיובי חוזר לסוף התור, שלילי נדחף למחסנית. מוציאים את ה-0, ומרוקנים את המחסנית לסוף התור (השליליים בסדר הפוך).

**שאלה 10**

**א.** מחפשים את הקוד ב-`codeList`; אם נמצא – לא עושים כלום. אחרת מוסיפים חוליה חדשה עם הקוד לראש הרשימה ומגדילים את `totalSum` ב-1.

**ב.** עוברים על `clients` ומשווים את `getId()` של כל לקוח ל-`id` (בעזרת `equals`); אם נמצא – יוצאים. אחרת יוצרים `new Client(id)` ומוסיפים אותו בחוליה חדשה לראש הרשימה.

**ג.** עוברים על `aparts`; לכל דירה שעירה שווה ל-`city`, מספר חדריה שווה ל-`numRooms` ומספר תמונותיה לפחות `numPictures` – מדפיסים את הקוד ומסמנים `found = true`. אם בסוף `found` עדיין `false` – מדפיסים "No match".

כל מבחני מה"ט עם פתרונות מלאים: www.guyai.co.il/mahat