בית ספר אפי ארזי למדעי המחשב

מבוא למדעי המחשב

מבחן גמר, 2023, מועד ב'

- זמן המבחן: שלוש שעות. לא תהיה הארכת זמן.
- יש לעבוד ביעילות. אם נתקעים בסעיף מסוים, מומלץ לעזוב אותו ולרוץ הלאה.
- חומר סגור. השימוש בסיכומי הרצאה, שקפים, מחשבונים או אינטרנט אסור (פרט לזום). כל החומר שתצטרכו כדי לענות על שאלות המבחן ניתן בדפי המבחן ובדפי העזר. אפשר להשתמש במילון (ספר מודפס, לא דיגיטלי).
- ענו על כל השאלות על דפי המבחן.
- אפשר לענות על כל שאלה בעברית או באנגלית, לפי בחירתכם.
- כתב היד שלכם חייב להיות קריא וברור. תשובות לא קריאות תקבלנה ציון 0.
- אם תרגישו צורך לעשות הנחה מסוימת כדי לענות על שאלה מסוימת, ניתן לעשות זאת, כל עוד ההנחה סבירה ומנוסחת בדיוק ובבהירות.
- אם אתם לא מסוגלים לתת תשובה מלאה לשאלה מסוימת, תנו תשובה חלקית.
  תשובה נכונה באופן חלקי תקבל ניקוד חלקי.
- אם אתם לא מצליחים מסיבה כלשהי לכתוב את הקוד הנדרש, תארו במילים את המימוש שהייתם רוצים לבצע (בעברית או באנגלית). אם התיאור יהיה מדויק ולעניין, תקבלו ניקוד חלקי.
- אם התבקשתם לכתוב קוד שאמור לפעול על קלט / ארגומנטים, אין צורך לכתוב קוד שבודק אם הקלט / ארגומנטים תקינים, אלא אם כן נאמר כך בשאלה במפורש.
- אין צורך לתעד את הקוד שתכתבו במבחן, אלא אם כן אתם רוצים לומר לנו משהו על הקוד שכתבתם.
- ניתן לכתוב פונקציות עזר (helper functions), אבל ממש אין צורך לעשות זאת.
- הקוד שתכתבו יישפט, בין היתר, לפי האורך, האלגנטיות, והיעילות שלו. תוכניות ארוכות או מסורבלות ללא צורך תקבלנה פחות נקודות, אפילו אם הן ממלאות את המשימה שהוגדרה בשאלה.
- לא יורדו נקודות על טעויות סינטקס טריוויאליות.
  לדוגמא, במקום לכתוב `System.out.print(x)`, אפשר לכתוב `print(x)`.
- כשכותבים קוד במבחן, מותר לקרוא לכל אחת מהמתודות שמוזכרות בדפי המבחן ובדפי העזר, אפילו אם לא מימשתם את המתודות האלה. אסור לקרוא למתודות שלא מוזכרות במבחן.

בהצלחה!

בדפי העזר יש תיעוד לכמה פונקציות ממחלקות שהשתמשנו בהן במהלך הקורס. הקוד שתכתבו במבחן יכול לקרוא לפונקציות האלה, לפי הצורך.

חלק א' – 36 נקודות (שאלות 1-3)

1. (14 נקודות)

נתון מערך דו-ממדי ריבועי mat המכיל מספרים שלמים (חיוביים, שליליים ואפסים) המקיים את התנאים הבאים:

1. כל שורה במערך ממויינת בסדר עולה ממש (אין מספרים זהים בשורה)
2. כל עמודה במערך ממויינת בסדר עולה ממש (אין מספרים זהים בעמודה)

כתבו פונקציה המקבלת כפרמטר מערך דו-ממדי כנ"ל, ומחזירה את מספר המספרים השליליים במערך.
לדוגמא, אם המערך mat מכיל את המספרים הבאים:

-90	-86	-55	-9
-89	-85	-16	0
-70	-60	-10	5
10	12	14	15

הפונקציה תחזיר את הערך 10, שכן יש 10 מספרים שליליים במערך.

בהינתן שמספר השורות והעמודות במערך הוא n, סיבוכיות הזמן הנדרשת היא לינארית O(n).
תשובה בסיבוכיות זמן גדולה מהדרש תזכה במעט נקודות.

חתימת הפונקציה היא:

```
public static int numOfNegativeNumbers (int [][] mat)
```

Extra page

2. (10 נקודות)

כתבו פונקציה **רקורסיבית** המקבלת מחרוזת str ומשתנה x מטיפוס int. הפונקציה תחזיר true במידה ו-x התווים הראשונים במחרוזת סימטריים ל-x התווים האחרונים במחרוזת, ו-false אחרת.

לדוגמא:

```
isSymmetric(“abdgutyrgdba”,4) // true
isSymmetric(“abdgutyrgdba”,2) // true
isSymmetric(“abca”,2) // false
isSymmetric(“aaa”,2) // true
isSymmetric(“aaaa”,5) // false
```

חתימת הפונקציה:

```
public static Boolean isSymmetricX(String str, int x)
```

3. (12 נקודות)

הגדרה: מספר שלם גדול מ-1 ייקרא מספר **k – כמעט ראשוני**, אם יש לו בדיוק k גורמים ראשוניים.

לדוגמא:

המספר 14 הוא 2-כמעט ראשוני, כי יש לו שני גורמים ראשוניים: 2*7
המספר 13 הוא 1-כמעט ראשוני, כי יש לו גורם ראשוני אחד: 13
המספר 100 הוא 4-כמעט ראשוני, כי יש לו ארבעה גורמים ראשוניים: 2*2*5*5

ממשו את הפונקציה `kAlmostPrime` המקבלת כפרמטר שני מספרים שלמים n>1 k>0 , ומחזירה true אם n הוא
k - כמעט ראשוני, ו-false אחרת.
החתימה של הפונקציה:

```
public static boolean kAlmostPrime(int n, int k)
```

Extra page

חלק ב'- 64 נקודות (שאלות 4-12) – OOP

השאלות הבאות מבוססות על המחלקות הבאות:

A. **Worker**: מתארת עובד ע"י שם name ומשכורת salary. name מטיפוס String, ו- salary מספר חיובי מטיפוס int.
B. **Manager**: הינו Worker, כלומר, יורש מ-Worker. ל-Manager יש שדה נוסף בשם workers מטיפוס LinkedList<Worker>, המחזיק את העובדים של המנהל. רשימת העובדים ממוינת לקסיקוגרפית לפי שם.
C. **Office** מכיל 3 שדות: name מטיפוס String , location מטיפוס String , ו-manager מטיפוס Manager.

- **בשאלות הבאות (4-12) תתבקשו לממש מתודות מהמחלקות Worker, Manager ו-Office.**

הAPI עבור המחלקות לעיל נמצאות במסמך API's, יחד עם ה-API עבור LinkedList<E> ו-ListIterator<E> .

שימו לב – ה-API של Node לא נתון לכם ולכן אינכם יכולים להשתמש בו.

בשאלה 4 תממשו מתודה מהמחלקה Worker.

4. (7 נקודות)

א. ממשו את המתודה raiseSalary(int amount). אם amount הינו מספר שלילי, על המתודה להחזיר את השגיאה IllegalArgumentException.

```
public void raiseSalary(int amount)
```

ב. איזו בעיה יכולה להיווצר אם עובד יקבל העלאה (אחת או יותר) שהסכום שלה גדול מאוד (מיליארדים)? כיצד ניתן לפתור את הבעיה?

בשאלות 5-7 תממשו מתודות מהמחלקה Manager

5. (5 נקודות)
הבנאי `Manager(String name, int salary, Worker[] workers)` מייצר מנהל בשם name, משכורת salary, ו-workers הם העובדים שעובדים ישירות תחתיו.
ממשו את הבנאי.

```
public Manager (String name, int salary, Worker[] workers)
```

6. (12 נקודות)

המתודה `getTotalSalaries()` מחזירה את סה"כ המשכורות של כל העובדים תחת המנהל (העובדים הישירים וגם לא הישירים), **לא כולל המנהל** עצמו.

המלצה: המימוש צריך לכלול רקורסיה ולולאה.

ממשו את המתודה:

```
public long getTotalSalaries()
```

7. (5 נקודות)
ממשו את המתודה equals במחלקה Manager.

```
public boolean equals(Object obj)
```

בשאלות 8-12 תממשו מתודות מהמחלקות Office ו-Manager

8. (14 נקודות)
המתודה getWorkersManager(Worker worker) מחזירה את המנהל הישיר של העובד worker. ממשו את המתודה.
המלצה: המימוש צריך לכלול רקורסיה ולולאה.

```
public Manager getWorkersManager(Worker worker)
```

בחרו אחת מהשאלות 9 או 10.

שימו לב, תצטרכו להשתמש **בשתי** מתודות אלו בהמשך.

9. (9 נקודות)
המתודה (addWorkerToManager (Worker worker, Manager manager מוסיפה עובד worker תחת מנהל manager. יש להוסיף את העובד לרשימת העובדים במיקום מסודר לקסיקוגרפי הנכון (לפי שמם). הינכם יכולים להניח שאין שני עובדים בעלי שם זהה. כחלק מהמימוש של מתודה זו, ממשו את המתודה addWorker השייכת למחלקה Manager.

שימו לב: הינכם יכולים להניח שהעובד והמנהל אינם זהים והעובד אינו מוקצה לשום מנהל אחר.

```
// Manager Class
public void addWorker(Worker worker)
```

```
// Office Class
public void addWorkerToManager (Worker worker, Manager manager)
```

10. (9 נקודות) – שימו לב יש לבחור את שאלה זו או שאלה 9

המתודה removeWorkerFromOffice מוציאה את העובד worker מהמשרד. הנכם יכולים להניח ש- worker הוא לא מנהל המשרד ואינו מנהל עם עובדים. כחלק מהמימוש של מתודה זו, ממשו את המתודהremoveWorker השייכת למחלקה Manager.

```
// Manager Class
public void removeWorker (Worker worker)
```

```
// Office Class
public void removeWorkerFromOffice(Worker worker)
```

11.(5 נקודות)
המתודה `getTotalSalaries()` מחזירה את סה"כ המשכורות במשרד, כולל המנהל.
ממשו את המתודה.

```
public long getTotalSalaries()
```

.

12.(5 נקודות)
המתודה moveWorkerBetweenManagers מעבירה את העובד ממנהל אחד לאחר.
ממשו את המתודה והשתמשו במתודות הקודמות.

```
public void moveWorkerBetweenManagers(Worker worker,Manager newManager)
```

Extra page

Extra page