

# מבני נתונים ואלגוריתמים

## חורף תשפ"ו

תרגיל בית תכנותי

להגשה עד ה – 29/01/2026 בשעה 23:59.

בהצלחה!

תרגיל זה מנוסח בלשון זכר מטעמי נוחות בלבד והוא מיועד לכל המגדרים.

מתרגל אחראי על התרגיל: ים סלונימסקי.

### הוראות:

- יש להגיש קובץ zip יחיד המכיל את כל קבצי ה – java שהשתמשתם בהם, כאשר השם של הקובץ הוא תעודות זהות של חברי הקבוצה המופרדים על ידי קו תחתון. למשל אם תעודות הזהות של שני השותפים הן ID1, ID2 אז השם של הקובץ יהיה ID1\_ID2.zip. שימו לב שבתוך ה – zip צריכים להיות **הקבצים עצמם ולא תיקיות נוספות**.
- ההגשה תתבצע רק ע"י אחד מבני הזוג למקום הייעודי באתר הקורס במודל.
- עליכם לוודא לפני ההגשה במודל כי הקוד שלכם מתקמפל ורץ בשרת Microsoft Azure שהוקצה לכם (הוראות מצורפות בקובץ בנפרד).
- מומלץ ואף רצוי להשוות את הפלט שלכם לפלט המופיע בקובץ master.txt המצורף לצורך בדיקות נכונות.
- זוג שהתרגיל שלו לא יתקמפל בשרת שהוקצה או יעוף בזמן ריצה ציונו בתרגיל יהיה 0.
- יש לכתוב קוד קריא ומסודר עם שמות משמעותיים למשתנים, למתודות ולמחלקות.
- יש להקפיד למלא את כל דרישות התרגיל (שימוש בייצוג נכון, סיבוכיות זמן וכו'). אי עמידה בדרישות התרגיל תגרור ציון 0.

## 1. כללי

קופת החולים "רפואית" זקוקה למערכת פשוטה ויעילה לניהול פעילות המרפאה שלה. במסגרת התרגיל, אתם, סטודנטים בקורס מבני נתונים ואלגוריתמים, התבקשתם לפתח מערכת שתסייע בניהול הרופאים הפעילים במרפאה ובמעקב אחר המטופלים הממתינים לכל רופא. המערכת תאפשר הוספה והסרה של רופאים במהלך היום, וכן רישום מטופלים לחדרי ההמתנה של הרופאים השונים. מטרתכם היא ליצור פתרון שידע להתמודד עם כמות גדולה של רופאים ומטופלים תוך שמירה על ביצועים אופטימליים.

## 2. פרטים

כאשר המרפאה נפתחת (אתחול המערכת) היא ריקה לחלוטין מרופאים וממטופלים. רופא מזוהה באמצעות מזהה ייחודי *doctorId*. כאשר רופא מגיע למרפאה נרצה לאפשר הכנסה שלו למערכת. מטופל מזוהה גם הוא באמצעות מזהה ייחודי *patientId*. כאשר מטופל מגיע למרפאה, הוא מגיע לפגישה עם רופא מסוים, לכן נרצה להכניס אותו לחדר ההמתנה עבור אותו הרופא.

ייתכן ומטופל יהיה מעוניין לעזוב את המרפאה לפני שנפגש עם הרופא ולכן יצא מחדר ההמתנה. נרצה לאפשר העברה של מטופל מחדר ההמתנה של רופא מסוים אל הרופא עצמו. מטופלים נכנסים לרופא אליו הגיעו באותו הסדר בו נכנסו לחדר ההמתנה שלו. כאשר הגיע זמנו של מטופל להיכנס לרופא, הוא למעשה יוצא מחדר ההמתנה של הרופא ומהמערכת כולה.

בנוסף נרצה לאפשר שליפה של נתונים הקשורים לרופאים, למטופלים ולמספר המטופלים הממתינים ברגע בחדרי ההמתנה (פירוט על אילו פעולות בדיוק נמצא בדרישות הממשק).

כאשר רופא מסיים את עבודתו, נרצה לאפשר הוצאה שלו מהמערכת. שימו לב שרופא יכול לעזוב את המערכת אך ורק אם חדר ההמתנה שלו ריק.

## 2.1. סימונים ודרישות

- בתרגיל זה נסמן את מספר הרופאים שנמצאים כרגע במערכת ב-  $D$  ואת מספר המטופלים שנמצאים כרגע במערכת ב-  $P$ .
- אלא אם נאמר אחרת, אתם **לא יכולים** להניח שהקלט לפעולות תקין. הגדרתו של קלט לא תקין נמצא תחת ההסבר אודות הפונקציה בהגדרת הממשק (3). במקרה והקלט לא תקין, יש לזרוק חריגה מסוג `IllegalArgumentException` **בלבד**.
- אסור לכם לייבא אף מחלקה או פונקציה שלא כתבתם, כלומר המילה השמורה `import` **לא יכולה להופיע** באף קובץ שתגישו. כמו כן השימוש במחלקה `System` אסור גם הוא.

## 3. ממשק

- את מבנה הנתונים שמייצג את המרפאה תבנו בתוך מחלקה פומבית בשם `ClinicManager` בתוך קובץ `ClinicManager.java`.
- למחלקה חייב להיות בנאי ברירת מחדל `ClinicManager()` שיאתחל את המערכת בסיבוכיות זמן  $O(1)$ .
- בקובץ ה- `ClinicManager.java` שמסופק לכם עם התרגיל, במחלקה `ClinicManager`, קיימים שני שדות קבועים לשימושכם:
  - `String MIN_ID`: מזהה ייחודי שמובטח לכם שכל מזהה שיתקבל במערכת יהיה גדול ממנו.
  - `String MAX_ID`: מזהה ייחודי שמובטח לכם שכל מזהה שיתקבל במערכת יהיה קטן ממנו.
- שימו לב שאם **תחליטו** להשתמש בשדות האלו, הם **חייבים להופיע** בקובץ `ClinicManager.java` בדיוק כפי שהם מופיעים בקובץ אותו סיפקנו לכם.

כמו כן, על המחלקה להכיל את הפעולות הבאות:

**void doctorEnter(String doctorId)**

מכניסה למערכת את הרופא בעל המזהה הייחודי `doctorId`.

סיבוכיות זמן:  $O(\log(D))$ .

- לא ניתן להכניס למערכת רופא שכבר נמצא בה.

**void** doctorLeave(**String** *doctorId*)

מוציאה מהמערכת את הרופא בעל המזהה הייחודי *doctorId*.

סיבוכיות זמן:  $O(\log(D))$ .

- לא ניתן להוציא מהמערכת רופא שאינו נמצא בה.
- לא ניתן להוציא מהמערכת רופא שחדר ההמתנה שלו אינו ריק.

**void** patientEnter(**String** *patientId*, **String** *doctorId*)

מכניסה לחדר ההמתנה של הרופא בעל המזהה הייחודי *doctorId*, את המטופל בעל המזהה הייחודי

*patientId*.

סיבוכיות זמן:  $O(\log(D) + \log(P))$ .

- לא ניתן להכניס למערכת מטופל שכבר נמצא בה.
- לא ניתן להכניס מטופל לחדר ההמתנה של רופא שאינו נמצא במערכת.

**String** nextPatientLeave(**String** *doctorId*)

מוציאה מחדר ההמתנה (ומהמערכת כולה) את המטופל הבא מחדר ההמתנה של הרופא בעל המזהה הייחודי

*doctorId* ומחזירה את המזהה הייחודי של אותו המטופל.

סיבוכיות זמן:  $O(\log(D) + \log(P))$ .

- לא ניתן להוציא מטופל מחדר ההמתנה של רופא שאינו נמצא במערכת.
- לא ניתן להוציא מטופל מחדר המתנה ריק.

**void** patientLeaveEarly(**String** *patientId*)

מוציאה מחדר ההמתנה (ומהמערכת כולה) את המטופל בעל המזהה הייחודי *patientId*.

סיבוכיות זמן:  $O(\log(D) + \log(P))$ .

- לא ניתן להוציא מהמערכת מטופל שאינו נמצא בה.

**int** numPatients(**String** *doctorId*)

מחזירה את מספר המטופלים שנמצאים כרגע בחדר ההמתנה של הרופא בעל המזהה הייחודי *doctorId*.

סיבוכיות זמן:  $O(\log(D))$ .

- לא ניתן לבדוק את מספר המטופלים בחדר המתנה של רופא שאינו נמצא במערכת.

**String** nextPatient(**String** *doctorId*)

מחזירה את המזהה הייחודי של המטופל הבא בחדר ההמתנה של הרופא בעל המזהה הייחודי *doctorId*.  
סיבוכיות זמן:  $O(\log(D))$ .

- לא ניתן להחזיר את המזהה הייחודי של המטופל הבא בחדר ההמתנה של רופא שאינו נמצא במערכת.
- לא ניתן להחזיר את המזהה הייחודי של המטופל הבא בחדר המתנה ריק.

**String** waitingForDoctor(**String** *patientId*)

מחזירה את המזהה הייחודי של הרופא שבחדר ההמתנה שלו נמצא כרגע המטופל בעל המזהה הייחודי *patientId*.  
סיבוכיות זמן:  $O(\log(P))$ .

- לא ניתן להחזיר את המזהה הייחודי של הרופא אליו ממתין מטופל שאינו נמצא במערכת.

**int** numDoctorsWithLoadInRange(**int** *a*, **int** *b*)

מחזירה את מספר הרופאים שכמות המטופלים שממתינים כרגע בחדר ההמתנה שלהם, נמצאת בטווח  $[a, b]$ .  
סיבוכיות זמן:  $O(\log(D))$ .

**int** averageLoadWithinRange(**int** *a*, **int** *b*)

מחזירה את ממוצע מספר המטופלים מבין כל חדרי ההמתנה שמספר המטופלים שנמצאים בהם כרגע, נמצא בטווח  $[a, b]$  – מעוגל כלפי מטה במקרה של שבר.  
סיבוכיות זמן:  $O(\log(D))$ .