תרגיל בית תכנותי להגשה

עד 8.4.2024 בשעה 23:59
בהצלחה!

תרגיל זה מנוסח בלשון זכר מטעמי נוחות בלבד והוא מיועד לכל המגדרים.
מתרגל אחראי על התרגיל: אורי מירז ועילי אבני

<u>הוראות:</u>

- יש להגיש קובץ zip יחיד כאשר השם של הקובץ הוא תעודות זהות חברי הקבוצה מופרדים על ידי קו תחתון למשל אם שני שותפים עם ID1, ID2 אז השם של קובץ ההגשה יהיה ID1_ID2.zip
- 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 נפרד.
- 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.

תיאור כללי:

לקראת האולימפיאדה הקרבה ובאה, הועד בישראל שאחראי על בחירת המתחרים זקוק למומחים במבני נתונים ואלגוריתמים ולכן ביקש מכם – סטודנטים מן המניין בפקולטה למדעי הנתונים וההחלטות שלוקחים את הקורס.

נבחרתם לעזור בבחירת אצן במקצוע "800 מ'". באופן מפורש, אתם מתבקשים לממש מבנה נתונים דינמי המסוגל לשמור את נתוני האצנים ולחשב מדדים על בסיס זמני הריצות שלהם באופן יעיל.

מבנה התחרות:

לכל אצן מוקצה מזהה ייחודי (אין להניח על המזהה שום דבר חוץ מכך שניתן להשוות בין המזהים).
כל אצן יבצע מספר מקצים, כאשר נרצה לשמור את ביצועיו בכל מקצה. יתכן כי בחלק מן המקצים התבצעה טעות במדידה ולכן נשמור לעצמינו את האפשרות לפסול ולמחוק ריצות מסוימות.

במסגרת התחרות, כל אצן ירוץ בנפרד בסדר שרירותי ולאחר כל מקצה נעדכן את תוצאתו.

בתרגיל בית זה אתם מתבקשים לממש בשפת **Java** את מבנה הנתונים הנ"ל באמצעות מחלקה בשם **Race.**

הערות חשובות לתרגיל:

- לכל אורך התרגיל נסמן ב-**n** את מספר האצנים וב-m_{runner} את מספר המקצים שאצן ביצע בעת הפעלת הפונקציה שאותה תממשו.
- אלא אם נאמר אחרת, **לא** ניתן להניח שהקלט תקין. במקרה והקלט אינו תקין (לדוגמא מחיקת אצן שלא קיים, מחיקת ריצה שקיימת, הוספת אצן שכבר קיים ועוד) יש לזרוק חריגה מסוג **IllegalArgumentException**.

עליכם ליצור קובץ בשם Race.java שבו תממשו מחלקה פומבית בשם Race שבה יהיה המבנה נתונים המאפשר את ניהול התחרות.

<u>על המבנה נתונים שלכם לתמוך בפעולות הבאות:</u>

- **חתימה:**
```
public void init()
```
תיאור הפונקציה: אתחול המרוץ.

סיבוכיות: $O(1)$

- **חתימה:**
```
public void addRunner(RunnerID id)
```
תיאור הפונקציה: הוספת אצן למבנה הנתונים בהינתן מזהה ייחודי.

סיבוכיות: $O(\log(n))$

- **חתימה:**
```
public void addRunToRunner(RunnerID id, float time)
```
תיאור הפונקציה: הוספת ריצה של אצן (ניתן להניח כי זמני הריצה של כל אצן ייחודיים)

סיבוכיות: $O(\log(\mathrm{n}) + \log(m_{runner}))$

- **חתימה:** `public void removeRunFromRunner(RunnerID id, float time)`

תיאור הפונקציה: מחיקת ריצה של אצן. ניתן להניח יחידות של הזמנים לכל אצן.

סיבוכיות: $O(\log(\mathrm{n}) + \log(m_{runner}))$

- **חתימה:** `public void removeRunner(RunnerID id)`

תיאור הפונקציה: מחיקת אצן (כולל מחיקה של כל הופעה שלו או של ריצה שלו ממבנה הנתונים).

סיבוכיות: $O(\log(n))$

- **חתימה:** `public float getMinRun(RunnerID id)`

תיאור הפונקציה: חישוב זמן הריצה המינימלי של אצן לפי המזהה שלו.

סיבוכיות: $O(\log(n))$

- **חתימה:** `public float getAvgRun(RunnerID id)`

תיאור הפונקציה: חישוב זמן הריצה הממוצע של אצן לפי המזהה שלו.

סיבוכיות: $O(\log(n))$

- **חתימה:** `public RunnerID getFastestRunnerAvg()`

תיאור הפונקציה: מציאת האצן עם זמן הריצה הממוצע הקטן ביותר.

סיבוכיות: $O(1)$

- **חתימה:** `public RunnerID getFastestRunnerMin()`

תיאור הפונקציה: מציאת האצן עם זמן הריצה המינימלי הקטן ביותר.

סיבוכיות:$O(1)$

- **חתימה:** `public int getRankAvg(RunnerID id)`

תיאור הפונקציה: מציאת ה*rank* של האצן לפי זמני הריצה הממוצעים של האצנים.

סיבוכיות: $O(\log(n))$

- **חתימה:** `public int getRankMin(RunnerID id)`

תיאור הפונקציה: מציאת ה*rank* של האצן לפי זמני הריצה המינימאליים של האצנים.

סיבוכיות: $O(\log(n))$

<u>הבהרות לגבי התרגיל</u>

- אם קיים אצן שהוספנו אך עוד לא התווספה לו ריצה או שכל הריצות שלו הוסרו – ניתן להניח שהממוצע והזמן המינימאלי הם Float.MAX_VALUE שניות.
- שבירת שוויון נעשות לפי הid של האצנים. כלומר, אם לשני אצנים יש אותו זמן ממוצע, האצן שיוחזר בgetFastestRunner (ובכל פונקציה אחרת שנדרש) יהיה האחד עם הid הקטן ביותר. ניתן להניח כי אפשר להשוות בין הid של האצנים.

<u>הסבר על הקבצים שקיבלתם</u>

1. Race.java – זה הקובץ שאתם כותבים בו את הקוד שלכם.

2. main.java – הקובץ שבו תרוץ בדיקת התרגיל.
3. RunnerID.java – קובץ שמכיל את המחלקה האבסטרקטית RunnerID.

<u>מבני ההגשה</u>

- יש להגיש קובץ zip אשר מכיל את הקובץ Race.java, RunnerID.java וכל קובץ אחר שהשתמשתם כדי לפתור את התרגיל.
- קוד אשר ישתמש בקבצים אחרים (לדוגמא בשביל מימוש מחלקות מסוימות) שלא יהיו בzip, יקבל 0.

<u>אילוצים:</u>

- הקוד אינו יכול להכיל import לשום מחלקה שלא מימשתם.
- אין להשתמש במחלקה System (אין לרשום בקוד שאתם יוצרים .System)