תרגיל בית תכנותי
להגשה עד 26.01.2023 בשעה 23:59
בהצלחה!

תרגיל זה מנוסח בלשון זכר מטעמי נוחות בלבד והוא מיועד לכל המגדרים.
מתרגל אחראי על התרגיל: ויסאם היגא

<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 מגישים 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 הנכון, סיבוכיות זמן וכו') אי עמידה בדרישות התרגיל תגרור ציון 0.

בעקבות האווירה וחווית המונדיאל, הטכניון החליט לקיים טורניר כדורגל בין הפקולטות כדי לדעת מי היא הפקולטה **הנחשבת** בכדורגל, אך בשונה מהמונדיאל הטכניון החליט שהזוכה במונדיאל היא הפקולטה עם הכי הרבה נקודות.

אז איך הטורניר מנוהל:

לכל פקולטה מוקצה מספר חיובי שלם המהווה מזהה ייחודי. בנוסף, לכל קבוצה יש שם ו קבוצת שחקנים שמורכבת מלפחות שני שחקנים ומקסימום 11 כאשר לכל שחקן יש מזהה ייחודי ושם. בנוסף, שחקן יכול להיות משויך לפקולטה מסוימת או להיות FREE AGENT. שימו לב שבניגוד לקבוצה פקולטית, אין חסם על כמות ה FREE AGENTS.

איך המשחקים מנוהלים:

משחקים נערכים בין שתי פקולטות. ניצחון שווה 3 נקודות; תיקו שווה נקודה אחת; הפסד שווה 0 נקודות.

שימו לב ייתכן שפקולטה כלשהי תשחק 3 משחקים ותהיה עם 9 נקודות ופקולטה אחרת תשחק 4 משחקים ותהיה עם 12 נקודות והיא תנצח את הטורניר למרות שהם לא שחקו אותו מספר של משחקים.

בתרגיל בית זה אתם מתבקשים לממש בשפת **Java** מבנה נתונים דינמי המאפשר ניהול הטורניר של הטכניון הנקרא **TechnionTournament**.

הערות חשובות לתרגיל:

- לכל אורך התרגיל הניחו כי **n** הוא מספר הקבוצות הנוכחי במערכת ו **m** הוא מספר השחקנים הנוכחי במערכת.
- לאורך כל השאלה ניתן להניח שנקלט של הפונקציות הוא תקין, כלומר אין צורך לבדוק תקינות קלט למשל: אין צורך ב לבדוק אם מזהה הפקולטה שמוסיפים אותה הוא חיובי ממש. וגם לא נבדוק אותם על זה 😉

עליכם ליצור עליכם ליצור קובץ בשם TechnionTournament.java ובו יש מחלקה פומבית בשם TechnionTournament ולה המתודות הפומביות הבאות:

בנוסף לזה אתם תקבלו שתי מחלקות Faculty, Player שנמצאות בקבצים Faculty.java ו Player.java כאשר:

המחלקה Faculty מכילה faculty_id, faculty_name

המחלקה Player מכילה player_id, player_name

תשתמשו במחלקות האלה כדי לקבל קלט וכדי להחזיר ערכים איפה שמצוין בפונקציות למטה.

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

```
public void init()
```

<u>תיאור הפונקציה:</u> בונה ברירת מחדל.

<u>סיבוכיות נדרשת:</u> $O(1)$

```
public void addFacultyToTournament(Faculty faculty)
```

<u>תיאור הפונקציה:</u> הוספת פקולטה חדשה לטורניר כאשר המזהה והשם של הפקולטה נמצאים בתוך faculty.

<u>סיבוכיות זמן:</u> $O(\log n)$

```
public void removeFacultyFromTournament(int faculty_id)
```

<u>תיאור הפונקציה:</u> הסרת פקולטה קיימת בטורניר שהמזהה שלה הוא faculty_id והפיכת כל השחקנים של הפקולטה ל Free Agent.

<u>סיבוכיות זמן:</u> $O(\log n + \log m)$

```
public void addPlayerToFaculty(int faculty_id,Player player)
```

<u>תיאור הפונקציה:</u> הפונקציה מוסיפה שחקן חדש עם מזהה player_id ושם player_name שנמצאות בתוך המחלקה player לקבוצת הפקולטה עם המזהה faculty_id.

<u>סיבוכיות זמן:</u>$O(\log n + \log m)$

```
public void removePlayerFromFaculty(int faculty_id, int player_id)
```

<u>תיאור הפונקציה:</u> הסרת השחקן עם המזהה player_id מן קבוצת הפקולטה faculty_id והשחקן הופך ל- Free Agent.

<u>סיבוכיות זמן:</u>$O(\log n + \log m)$

```
public void playGame(int faculty_id1, int faculty_id2, int winner,
                     ArrayList<Integer> faculty1_goals, ArrayList<Integer> faculty2_goals)
```

<u>תיאור הפונקציה:</u> הפונקציה מתארת משחק בין פקולטה עם המזהה faculty_id1 לבין פקולטה עם המזהה faculty_id2 . תוצאת המחשק נמצאת ב winner:

1. אם הערך שלו 1 אז הפקולטה עם faculty_id1 ניצחה
2. ואם הערך שלו הוא 2 אז הפקולטה עם המזהה faculty_id2 ניצחה,
3. ואם winner מכיל אפס זאת אומרת ש התוצאה של המשחק היא תיקו.

בנוסף לזה אתם מקבלים מערכים של מזהים של השחקנים שהבקיעו לכל קבוצה.

שימו לב ייתכן כפוליות במערכים faculty_goals כי ייתכן ש שחקן הבקיע יותר משער אחד למשחק.

<u>סיבוכיות זמן:</u>$O(\log n + k \cdot \log m)$ במקרה הגרוע, כאשר k כמות השערים במשחק, כלומר אורך של faculty1_goals ו faculty2_goals.

```
public void getTopScorer(Player player)
```

<u>תיאור הפונקציה:</u> הפונקציה מחזירה את מזהה ושם השחקן שהבקיע את מספר השערים הגדול ביותר לתוך המחלקה

(תסתכלו בקובץ Player.java כדי לדעת איך מעדכנים את המחלקה Player).

השתמשו בשובר שוויון שכתוב למטה בצבע צהוב.

<u>סיבוכיות זמן:</u>$O(1)$

```
public void getTopScorerInFaculty(int faculty_id, Player player)
```

<u>תיאור הפונקציה:</u> הפונקציה מחזירה את מזהה ושם השחקן שהבקיע את מספר השערים הגדול ביותר בפקולטה שהמזהה שלה הוא faculty_id לתוך המשתנה המחלקה player

(תסתכלו בקובץ Player.java כדי לדעת איך מעדכנים את המחלקה Player).

השתמשו בשובר שוויון שכתוב למטה בצבע צהוב.

<u>סיבוכיות זמן:</u>$O(\log n)$

```
public void getTopKFaculties(ArrayList<Faculty> faculties, int k, boolean ascending)
```

<u>תיאור הפונקציה:</u> הפונקציה מקבלת קבוע k ומשתנה בוליאני ascending ומזינה את המזהים והשמות של k הפקולטות עם הכי הרבה נקודות לתוך המערכים הנתונים בקלט .

אם $ascending = true$ אז צריך להחזיר את הפקולטות בסדר עולה כלומר המקום האחרון במערך תהיה הקבוצה עם הכי הרבה נקודות, אחרת בסדר יורד

(תסתכלו בקובץ Faculty.java כדי לדעת איך מעדכנים ומייצרים את המחלקה Faculty).

השתמשו בשובר שוויון שכתוב למטה בצבע צהוב.

<u>סיבוכיות זמן:</u>$O(k)$ כאשר k הוא הקבוע הנתון בפונקציה.

```
public void getTopKScorers(ArrayList<Player> players, int k, boolean ascending)
```

<u>תיאור הפונקציה:</u> בדומה לפונקציה קודמת, אך עכשיו מחזירים את המזהים והשמות של k השחקנים עם הכי הרבה שערים.

(תסתכלו בקובץ Player.java כדי לדעת איך מעדכנים ומייצרים את המחלקה Player).

השתמשו בשובר שוויון שכתוב למטה בצבע צהוב.

<u>סיבוכיות זמן:</u>$O(k)$ כאשר k הוא הקבוע הנתון בפונקציה.

```
public void getTheWinner(Faculty faculty)
```

<u>תיאור הפונקציה:</u> הפונקציה מחזירה את המזהה והשם של הפקולטה המנצחת (עם הכי הרבה נקודות) לתוך המחלקה faculty.

(תסתכלו בקובץ Faculty.java כדי לדעת איך מעדכנים את המחלקה Faculty).

השתמשו בשובר שוויון שכתוב למטה בצבע צהוב.

<u>סיבוכיות זמן:</u>$O(1)$

עליכם לחשוב איך מחלקה תשתמש במחלקה אחרת, ואולי להגדיר מחלקות, משתנים ומתודות נוספות כרצונכם.

הסבר חשוב לגבי שבירת שוויון:

- אם לשני שחקנים יש אותו מספר שערים ואתם נדרשים להחזיר אחד מן השחקנים אז מחזירים השחקן עם המזהה הקטן.
לדוגמה: אם יש לנו במערכת שני שחקנים עם המזהים 1,2 ושניהם עם אותו מספר שערים והוא המקסימלי בטורניר אז במידה ונקראת הפונקציה **getTopScorer(...)** אז מחזרים את השחקן עם המזהה 1

- אם לשתי קבוצות יש אותו ניקוד והתבקשתם להחזיר אחת מהם אז תחזירו את הקבוצה עם המזהה הקטן
לדוגמה: אם יש לנו שתי קבוצות עם המזהים 1,2 ושתי הקבוצות נמצאות עם הכי הרבה נקודות והם שווים, במידה ונקראת הפונקציה **getTheWinner(...)** אז צריך להחזיר את הקבוצה עם המזהה 1.

<u>הסבר על הקבצים שקיבלתם:</u>

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

2. Tournament.java – הקובץ מכיל את החתימות של הפונקציות שאתם צריכים לממש.
נא לא לשנות שום דבר בקובץ הזה.

3. Player.java - הקובץ מכיל מימוש של המחלקה Player שתשתמשו בה לאורך התרגיל כדי לקבל והחזיר ערכים מהפונקציות. נא לא לשנות שום דבר בקובץ הזה.
נא לא לשנות שום דבר בקובץ הזה.

4. Faculty.java - הקובץ מכיל מימוש של המחלקה Faculty שתשתמשו בה לאורך התרגיל כדי לקבל והחזיר ערכים מהפונקציות. נא לא לשנות שום דבר בקובץ הזה.
נא לא לשנות שום דבר בקובץ הזה.

5. Main.java – מכיל טסט קטן על איך הטסטים ייראו, אבל כמובן שתהיה הגבלת זמן על הפונקציות, אבל בטסט הנתון אין התייחסות לזה, המטרה שלו היא שתדעו איך הטסטים נראים וכל מי שרוצה לכתוב טסטים לעצמו אז הוא ידע איך, בנוסף לזה אתם מקבלים בדיקה פשוטה שהקוד שלכם עובד.

<u>הערה</u>: אתם יכולים להוסיף קבצים משלכם אבל אל תשכחו להגיש אותם.

<u>אילוצים:</u>

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

<u>המלצות:</u>

1. אל תשאירו את הבדיקה בשרת לרגע האחרון. ייתכן והקוד לא יתקמפל בשרת ותצטרכו לתקנו לפני ההגשה.
2. תכננו את המבנה שלכם ואז תתחילו לממש, תראו מה אתם צריכים ומה אתם לא צריכים זה יחסוך לכם בזמן, כי התרגיל אינו פשוט ואינו קצר.