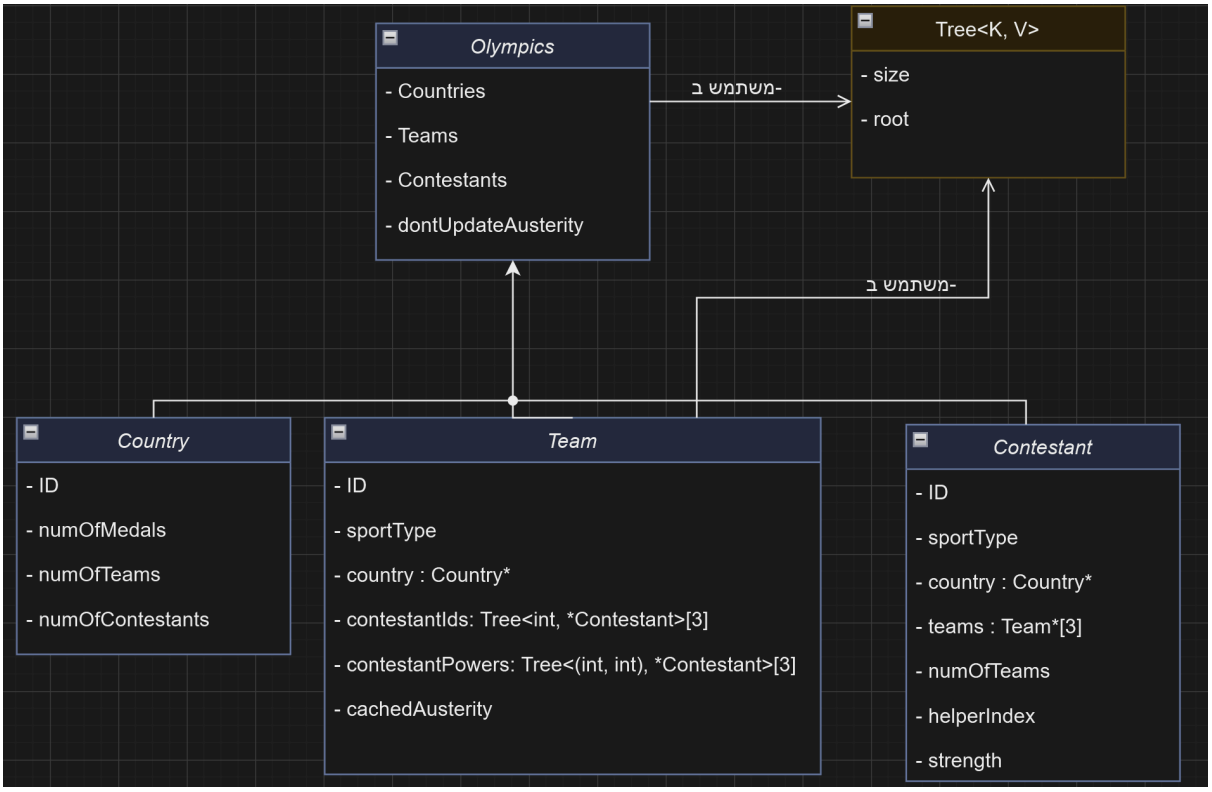
מבני גיליון רטוב 1 חלק יבש

יונתן רייכר	יונתן שטרנברג
214762718	214762338

מבני הנתונים:



פונקציות:

שם הפונקציה	מימוש	סיבוכיות
olympics_t()	שדות המבנה (מדינות, נבחרות, מתחרים) יאותחלו לעצים ריקים	O(1)
Virtual ~olympics_t()	נשחרר את כל צמתי העצים שלנו - סה"כ n+m+k צמתים	O(n+k+m)
StatusType add_country(int countryId, int medals)	אם countryId<=0 או medals<0 נחזיר INVALID_INPUT ניצור מדינה חדשה עם מספר המדליות הנתון ונוסיף אותה לעץ המדינות - log k אם נתקלנו במדינה עם אותו מזהה נחזיר FAILURE.	O(log k)
StatusType remove_country(int countryId)	אם countryId<=0 נחזיר INVALID_INPUT נחפש את המדינה בעץ המדינות - log k אם לא נתקלנו במדינה עם אותו מזהה או ששדה הקבוצות אינו 0 או שדה המשתתפים אינו 0 נחזיר FAILURE. אחרת נמחק אותה ונחזיר SUCCESS.	O(log k)
StatusType add_team(int teamId, int countryId, Sport sport)	אם countryId<=0 או teamId<=0 נחזיר INVALID_INPUT נבדוק שקיימת ארץ כזו ע"י חיפוש בעץ הארצות - log k אם הארץ אינה קיימת נחזיר FAILURE, אחרת, נגדיל באחד את שדה numOfTeams שלה. ניצור קבוצה חדשה ונוסיף אותה לעץ הקבוצות - log m נאתחל לה מצביע למדינה וכל עצייה יהיו ריקים וכוחה 0. אם נתקלנו בקבוצה עם אותו מזהה נחזיר FAILURE.	O(log m+log k)

StatusType remove_team(int teamId)	אם teamId<=0 נחזיר INVALID_INPUT נחפש את הקבוצה עם המזהה הנתון - log m אם לא מצאנו את הקבוצה או שקיימים מתחרים בקבוצה (לפי האם עצי ה-id שלה ריקים) נחזיר FAILURE, אחרת, נקטין באחד את שדה הnumOfTeams של המדינה ששמורה בנבחרת ונמחק את הנבחרת.	O(log m)
StatusType add_contestant(int contestantId, int countryId, Sport sport, int strength)	אם contestantId <= 0 או countryId <= 0 או strength < 0 אז נחזיר INVALID_INPUT. נחפש את המתחרה ואת המדינה. פעולות אלו הן log n ו-log k בהתאמה. במקרה והמדינה לא נמצאת, או שהמתחרה *כן* נמצא אז נחזיר FAILURE. אחרת ניצור מתחרה חדש, נאתחל לו מצביע למדינה, ניתן לו את ה-id שלו ונוסיף אותו לעץ המתחרים במקום ה-id. מערך הקבוצות שלו יהיה ריק ונאתחל את הספורט שלו לספורט הנתון. נגדיל את numOfContestants של המדינה באחד.	O(log n + log k)
StatusType remove_contestant(int contestantId)	אם contestantId <= 0 נחזיר INVALID_INPUT. נחפש את המתחרה בעץ המתחרים - log n אם הוא לא נמצא, או שהוא נמצא אך יש במערך שלו איבר (הוא רשום לנבחרת) אז נחזיר FAILURE. אחרת נוציא את המתחרה מעץ המתחרים לפי ID, נקטין את מספר המתחרים במדינה של המתחרה ונחזיר SUCCESS.	O(log n)
StatusType add_contestant_to_team(int teamId, int contestantId)	אם contestantId<=0 או teamId<=0 נחזיר INVALID_INPUT נחפש את הנבחרת בעץ הנבחרות - log m ונחפש את המתחרה בעץ המתחרים - log n אם הנבחרת או המתחרה לא קיימים, או שהמתחרה נמצא כבר בנבחרת או ששדות sportType/country לא תואמים בין הנבחרת למתחרה או שלמתחרה יש כבר 3 נבחרות שהוא משתתף בהן (נבדוק בעזרת שדה numOfTeams), נחזיר FAILURE. אחרת, נגדיל את מספר הקבוצות בהן משתתף המתחרה, ונוסיף את הקבוצה למערך הקבוצות של המתחרה. O(1) נוסיף גם מצביע אל המשתתף לעץ id של הקבוצה O(log n) - אם ה-id קטן מה-id המקסימלי של העץ הראשון - נוסיף לעץ הראשון - אחרת: אם קטן מהמקסימלי בשני, נוסיף לשני - אחרת נוסיף לשלישי במידת הצורך, נזיז משתתפים בין עצי id כדי לשמר את הסדר של העצים (נזיז את הקטן של השני לראשון, או הגדול של הראשון לשני וכן הלאה) כל משתתף שהוסף/הוזז יכנס לעץ הכח המתאים שלו (ויוצא מקודמו) נחשב את שדה האוסטריטי מחדש אם לא היה ניתן ליצור מתחרה חדש, או לא היה ניתן להוסיף אותו לעץ, נחזיר ALLOCATION_ERROR. אחרת SUCCESS	O(log n+log m)
StatusType remove_contestant_from_team(int teamId, int contestantId)	אם contestantId<=0 או teamId<=0 נחזיר INVALID_INPUT נחפש את הנבחרת בעץ הנבחרות - log m ונחפש את המתחרה בעץ המתחרים - log n אם הנבחרת או המתחרה לא קיימים, או שלמתחרה לא מופיעה הנבחרת במערך הנבחרות, נחזיר FAILURE.	O(log n+log m)

	אחרת, נקטין ב-1 את מספר הקבוצות בהן משתתף המתחרה, ונוציא את המצביע לקבוצה מהמערך. בנוסף, נוציא את את המשתתף מעץ הID ועץ הכח שלו, ובמידת הצורך נעביר משתתף בין עצים ע"מ לשמור על סדר גודל זהה בין העצים. נעדכן את שדה האוסטריטי מחדש	
StatusType update_contestant_strength(int contestantId , int change)	אם contestantId<=0 נחזיר INVALID_INPUT נחפש את המתחרה בעץ המתחרים - log n אם אין מתחרה עם מזהה contestantId או שהכח שלו אחרי השינוי קטן מ0 נחזיר FAILURE. אחרת, נעדכן את כוחו. בנוסף, נמצא את הנבחרות של המתחרה (בעזרת שדה הנבחרות שלו) בעץ המתחרים, לכל היותר 3*log m ולכן - log m נחפש בעצי הכוחות של הנבחרות את המתחרה - לכל היותר 3log n/3, כלומר - log n נמחק אותו מהעץ ונוסיף אותו שוב לאותו העץ עם הכוח המעודכן כפי שתיארנו מעלה (ונעדכן את האוסטריטי)	O(log n+log m)
output_t < int > get_strength(int contestantId)	אם contestantId<=0 נחזיר INVALID_INPUT נחפש את המתחרה בעץ המתחרים - log n אם לא מצאנו את המתחרה נחזיר FAILURE אחרת, נחזיר את ערך שדה הכוח שלו.	O(log n)
output_t < int > get_medals(int countryId)	אם countryId<=0 נחזיר INVALID_INPUT נחפש את המדינה בעץ המדינות - log k אם לא נמצאה מדינה כזו, נחזיר FAILURE אחרת, נחזיר את הערך השמור בשדה numOfMedals של המדינה.	O(log k)
output_t < int > get_team_strength(int teamId)	אם ה-id קטן או שווה לאפס נחזיר INVALID_INPUT נחפש את הקבוצה בעץ הקבוצות - פעולה זו היא log m אם לא נמצא - נחזיר FAILURE אחרת, נלך לשלושת עצי הכח, ונחזיר את הסכום של המפתחות המקסימליים שלהם O(1)	O(log m)
StatusType unite_teams(int teamId1, int teamId2)	אם 0 => teamId1 או 0 => teamId2 או שהם שווים זה לזה נחזיר INVALID_INPUT. נחפש את הקבוצות בעץ הקבוצות - log m אם אחת מהן לא נמצאה, או שהן שתיהן נמצאו אך הן ממדינות שונות או בספורט שונה, נחזיר FAILURE. אחרת, נפעל בצעדים הבאים: 1. נכתוב את כל העצים של הקבוצות למערכים עם שימור הסדר הפנימי של האיברים בעצים (O('n)). בכל קבוצה, 3 העצים של id ירשמו למערך אחד לפי הסדר ו3 עצי הכח ירשמו למערכים שונים 2. ניצור 5 מערכים חדשים נוספים ריקים בגודל n' 3. נכתוב את המשתתפים באופן מסודר לפי id למערך הראשון: O('n) a. נבדוק האם ה-id הראשון במערך של קבוצה 1 גדול או קטן מזה של קבוצה 2 b. את הקטן יותר מבינהם נוציא ממערכו ונדלג עליו ונשים במערך החדש הראשון במקום הריק הבא	O(log m + $n_{Team1ID}$ + $n_{Team2ID}$)

	(O(1)) אם האלמנט הקודם שווה אליו, אז בעצם לא נעשה את הפעולה הזו כדי שלא יהיו לנו כפילויות במערך c. נסמן את שדה האינדקס שלו באינדקס שלו במערך הנל. d. בין אם הוא בא מקבוצה 1 או 2, נמחק את קבוצה 2 מרשימת הקבוצות שלו ונדאג שקבוצה 1 מופיעה. e. נחזור חלילה עד שכולם נכתבו ומערכי ה-id של שני הקבוצות ריקים ומערך ה-id הגדול מלא 4. כעת נעבור על התהליך הנ"ל עם מערכי הכח, עם 6 מערכי מקור במקום 2 עצים. נכתוב את כולם למערך החדש השני. גם כן O(n'). הפעם לא נשנה את שדה האינדקס או את הקבוצות 5. נעבור על כל איבר במערך החדש השני: a. אם שדה האינקס שלו קטן מ-n/3 אז הוא יכנס למערך ה-3. אחרת, אם הוא קטן מ-2n/3 אז הוא יכנס למערך 4. אחרת הוא יכנס למערך 5. ח פה הוא מספר האיברים במערך id הגדול. 6. כעת ניצור עץ מכל שליש של מערך ה-id הגדול, וניצור עץ מהמערכי 3, 4 ו-5. 7. העצים האלו יחליפו את העצים של הקבוצה הראשונה 8. נעדכן את שדה האוסטריטי של קבוצה 1 9. נמחק את קבוצה 2 מעץ הקבוצות וננקה את כל הזיכרון הנותר (המערכים וכו'..) * סימון: $n' = n_{team1ID} + n_{team2ID}$	
O(log m)	אם teamId1<=0 או teamId2<=0 או teamId1==teamId2 נחזיר INVALID_INPUT. נחפש את שתי הנבחרות בעץ הנבחרות - log m אם לא מצאנו את הנבחרות או ששניהן מתחרות בסוגי ספורט שונים נחזיר FAILURE. נחשב את הניקוד של הקבוצות ונשווה אותו, אם לאחת הקבוצות יש ניקוד גבוהה יותר נוסיף למדינה שלה (בעזרת שדה המצביע למדינה) מדליה אחת.	**StatusType play_match(int teamId1, int teamId2)**
O(log m)	אם ה-id קטן או שווה לאפס נחזיר INVALID_INPUT נחפש את הקבוצה בעץ הקבוצות - פעולה זו היא log m אם לא נמצא - נחזיר FAILURE אם בקבוצה יש פחות משלושה משתתפים גם כן נחזיר FAILURE אחרת, נחזיר את ערך שדה האוסטריטי השמור בקבוצה. <u>הסבר חישוב שדה אוסטריטי:</u> אם יש פחות מ-6 משתתפים בקבוצה, בהכרח האוסטריטי הוא 0. אחרת: נעבור על כל האפשרויות לבחור 3 קבוצות. מכל קבוצה ננסה להוציא את המשתתף המקסימלי או את המשתתף המינימלי לכל אפשרות כזו נחשב את הכח של הקבוצה, ונחזיר את כל המשתתפים. כמות אפשרויות זו היא סופית לכן סך הכל הסיבוכיות היא O(log n). דרישת הסיבוכיות אז מתקיימת כי חישוב זה נעשה רק בפונקציות שהן בסיבוכיות זמן גדולה או שווה.	**output_t < int > austerity_measures(int teamId)**

- n - מספר מתחרים, m - מספר נבחרות, k - מספר מדינות
- בכל מקרה של בעיה בהקצאה/שחרור זיכרון נחזיר ERROR_ALLOCATION
- בכל מקרה של הצלחה נחזיר SUCCESS
- לאף פונקציה אין דרישת סיבוכיות מקום ולכן באופן ריק כל פונקציה מקיימת את דרישת סיבוכיות המקום
- מבנה הנתונים עצמו מקיים את סיבוכיות המקום כי כל מדינה וקבוצה נשמרים פעם אחת ואפילו שכל קבוצה יכולה להכיל כמות מצביעים עד לכמות המתחרים, כל מתחרה יכול להופיע בעד 3 קבוצות