מבני נתונים 1 – 234218
שרה גריפית – ת"ז: 341312304
ליאור בר יוסף – ת"ז: 207022443

תרגיל רטוב 1 – החלק היבש

שרה גריפית – ת"ז: 341312304
ליאור בר יוסף – ת"ז: 207022443

תיאור מבנה הנתונים:

עבור האפליקציה של פיפ"א, נבנה מבנה נתונים המורכב מארבע עצי AVL מרכזיים:

- **m_teamsByID** – כל הקבוצות שמשתתפות בגביע העולם, מסודרים לפי מזהה הקבוצה.
- **m_qualifiedTeams** – כל הקבוצות המשתתפות בגביע העולם אשר כשרים לשחק (בעלי לפחות 11 שחקנים ולפחות שוער אחד).
- **m_playersByID** – כל השחקנים שמשתתפים בגביע העולם, מסודרים לפי מזהה השחקן.
- **m_playersByScore** – כל השחקנים שמשתתפים בגביע העולם, מסודרים לפי ה-score של השחקן: השחקן עם ה-score הגבוה ביותר הוא עם מספר השערים הגדול ביותר. במקרה של שוויון, מספר הכרטיסים הנמוך ביותר. ובמקרה של שוויון, מזהה השחקן.

בגביע העולם יש גם משתנה נוסף הסוכם את כמות השחקנים שמשתתפים בכל גביע העולם, ובכן מצביע לשחקן עם ה-score הגבוה ביותר.

נוסיף שתי מחלקות: Player ו-Team אשר מכילים את הפרטים של השחקנים ושל הקבוצות כדורגל בגביע העולם.

- **מחלקת Player** תכיל את כל הפרטים הבסיסיים של אותו השחקן: מזהה שחקן, מספר המשחקים שהשחקן שיחק (בנפרד למשחקים ששיחק עם קבוצתו הנוכחית), מספר השערים שהבקיע, מספר הכרטיסים שקיבל והאם יכול לשחק כשוער. בנוסף יכיל מצביע לקבוצה בה משחק, ושני מצביעים לשחקנים שהכי "קרובים" לשחקן מבחינת score (אחד המצביע לשחקן הסמוך עם score גבוהה יותר והשני המצביע לשחקן הסמוך עם score נמוך יותר).
- **מחלקת Team** תכיל את הפרטים הבסיסיים של אותה קבוצה: מזהה קבוצה, מספר הנקודות של הקבוצה, מספר השחקנים בקבוצה, מספר השוערים בקבוצה, סה"כ השערים ששחקני הקבוצה הבקיעו, סה"כ כרטיסים ששחקני הקבוצה קיבלו ומספר המשחקים שהקבוצה שיחקה. בנוסף יש שני מצביעים לקבוצות עם מזהים סמוכים (אחד המצביע לקבוצה עם המזהה הסמוך הגדול יותר, והשני המצביע לקבוצה עם המזהה הסמוך הקטן יותר).
המצביעים הללו יצביעו לקבוצה (ולא nullptr) רק במידה והקבוצה <u>כשרה</u> למשחק (בעלת 11 שערים לפחות ולפחות שוער אחד). כלומר, נוצרת שרשרת של קבוצות כשרות למשחק (ובשאר הקבוצות המצביעים יצביעו ל-nullptr).
  - בנוסף לפרטים הבסיסיים, בקבוצה נחזיק שני עצי AVL – אחד של כל שחקני הקבוצה מסודרים לפי מזהה שחקן, והשני של כל שחקני הקבוצה מסודרים לפי ה-score של שחקני הקבוצה, המבוסס על כמות השערים שהבקיעו + הכרטיסים שקיבלו + מזהה השחקן (במקרה של שוויון), בדומה ל-m_playersByScore של world_cup_t.

נשים לב כי נוצר <u>עותק אחד בלבד</u> של כל שחקן וקבוצה המשתתפים בגביע העולם, והעצים/המצביעים השונים מצביעים לאותם העותקים.

<u>סיבוכיות מקום:</u> בכל פעולה, אין הוספה של סיבוכיות מקום מעבר לנדרש לכלל התרגיל. לכן סיבוכיות המקום נשאר O(n+k), כאשר n הוא כמות השחקנים ו-k הוא כמות הקבוצות בגביע העולם.

להלן רשימת הפעולות הנדרשות לממשק והסבר למימושם והסיבוכיות שלהם:

world_cup_t –

נאתחל את ארבעת עצי הAVL בעזרת קריאה לconstructor-ים שלהם, ונאתחל את המצביע לשחקן עם ה-score הגבוה ביותר ל-nullptr. כמו כן, נאתחל את המשתנה שסופר את כל השחקנים המשתתפים בגביע העולם ל-0. במקרה של

<u>סיבוכיות זמן:</u>
נאתחל את כל המשתנים ל-null או לערכים דיפולטים. עבור העצים – עץ ריק מכיל רק node אחד, וסיבוכיות הזמן ליצירתו הוא O(1). באופן דומה גם המשתנים ה-int-ים והמצביעים הנוספים. ולכן סיבוכיות הזמן הוא **O(1)**.

מבני נתונים 1 – 234218
שרה גריפית – ת"ז: 341312304
ליאור בר יוסף – ת"ז: 207022443

~world_cup_t –

ראשית נשחרר את ה-data של כל השחקנים והקבוצות בגביע העולם. נעשה זאת ע"י שחרור הזכרון של ה-data בעצים m_teamsByID ו-m_playersByID שמכילים את כל הקבוצות וכל השחקנים.
לאחר מכן, ה-destructor-ים הדיפולטים של העצים השונים יקראו, וישחררו את הזיכרון של ה-nodes של העצים.
בנוסף, יקראו ה-destructor-ים הדיפולטים למצביע של השחקן עם ה-score הכי גבוה ובכן למשתנה int הסופר את כמות סך השחקנים.

<u>סיבוכיות זמן:</u>
כדי לשחרר את ה-data, נעבור על כל הקבוצות והשחקנים שקיימים בגביע העולם, ונשחרר את הזיכרון. זהו נעשה בסיבוכיות O(n+k) כאשר n הוא כמות השחקנים בגביע העולם, ו-k הוא כמות הקבוצות בגיע העולם. בשחרור הזכרון של הקבוצות, נעשה גם הריסה של עצי השחקנים של הקבוצות:

- בתוך כל קבוצה יש שני עצים של השחקנים המשחקים באותה קבוצה. כל שחקן משחק בקבוצה בודדת, לכן סך כל השחקנים בעצים הללו <u>בכל</u> הקבוצות יחד הוא 2n (n בכל סוג עץ).
- לכן בהריסת כלל השחקנים, נעבור על עוד 2n nodes בעצי השחקנים, ולכן סיבוכיות הזמן הוא = O(2n) O(n).

לאחר מכן, יקראו ה-destructor-ים הדיפולטים של העצים השונים, אשר עוברים על כל ה-nodes בגביע העולם:

- יש לנו שני עצים של שחקנים, ולכן עבור n שחקנים, יש לכל עץ מקסימום n nodes. לכן סיבוכיות הזמן של מעבר על העצים הללו הוא O(2n) = O(n).
- יש לנו שני עצים של קבוצות, ולכן עבור k קבוצות, יש לכל עץ מקסימום k nodes. לכן סיבוכיות הזמן של מעבר על העצים הללו הוא O(2k) = O(k).

לכן סיבוכיות הזמן סך הכל הוא **O(n+k)** = O(7n + 3k) = O(n+k+2n+2k+4n)**.**

add_team –

בפעולה זו נוסיף קבוצה לגביע העולם. ראשית נוודא את תקינות הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה קבוצה הקטנה או שווה ל-0 או מספר הנקודות של הקבוצה שלילי.
כמו כן, נוודא כי לא קיים כבר קבוצה עם המזהה הזה, אחרת נחזיר FAILURE. נקצה זיכרון עבור הקבוצה החדשה, ונחזיר ALLOCATION_ERROR במקרה של כישלון בהקצאת הזיכרון.
נוסיף את הקבוצה לעץ AVL שמרכז את כל הקבוצות המסודרות לפי מזהה קבוצה – m_teamsByID. נשים לב כי בהוספת קבוצה חדשה, מספר השחקנים בה הוא 0 ולכן הקבוצה לא מתווספת לעץ AVL של m_qualifiedTeams והמצביעים לקבוצות הסמוכות לא מתעדכנים. בסוף נחזיר SUCCESS.

<u>סיבוכיות זמן:</u>
נעשה חיפוש לפי מזהה קבוצה בעץ m_teamsByID AVL כדי להכניס את הקבוצה החדשה, בסיבוכיות של O(log(k)). לאחר ההכנסה, נבצע גלגולים בעץ כדי לאזן אותו בחזרה, בסיבוכיות של O(1) לגלגול כפי שנלמד בהרצאה. נעבור על כל ה-nodes במעלה העץ (מקסימום log(k) nodes) עד השורש בעדכון גובה העץ וה- balance factor, ולכן איזון העץ הוא בסיבוכיות O(log(k)). לכן סך סיבוכיות הזמן של הפעולה היא **O(log(k))** = O(log(k)*2)**.**

remove_team –

ראשית נוודא את תקינות הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה שחקן קטן או שווה ל-0.
נחפש בעץ AVL של הקבוצות המסודרות לפי מזהה קבוצה - m_teamsByID, ונחזיר FAILURE במקרה שקיימים עדיין שחקנים בקבוצה, או שהקבוצה לא קיימת.
אחרת, נסיר את הקבוצה על ידי הסרת הnode בעץ m_teamsByID AVL (נשים לב כי אם לקבוצה אין שחקנים, אז הוא לא קיים בעץ m_qualifiedTeams ולכן אין צורך להסיר אותו משם). נשחרר את הזכרון של הקבוצה הזו, ובסוף נחזיר SUCCESS.

<u>סיבוכיות זמן:</u>
בפעולה זו אנו מחפשים את הקבוצה המבוקשת פעמיים נפרדות: פעם אחת לבדוק אם קיימים שחקנים בקבוצה – חיפוש בעץ m_teamsByID, ופעם שנייה כדי להסיר את הקבוצה מ-m_teamsByID. בשני המקרים, סיבוכיות הזמן

של החיפוש הוא O(log(k)) כי יש מקסימום k קבוצות בעץ הזה. ההסרה מהעץ כוללת גלגולים בשיטה שנלמדה בהרצאה – בסיבוכיות O(1) לכל גלגול. בפוטנציאל יש log(k) גלגולים כי עוברים על כל הnodes במעלה העץ.
לכן סך סיבוכיות הזמן הוא **O(log(k))** = O(log(k)*3).

– add_player

ראשית נוודא את תקינות הקלט, ואם הוא אינו תקין אז נחזיר INVALID_INPUT.
כמו כן, נוודא כי לא קיים כבר שחקן עם המזהה הזה וקיימת קבוצה עם המזהה הניתן, אחרת נחזיר FAILURE.
ניצור את השחקן עם הנתונים מהקלט חוץ מכמות המשחקים שהשחקן שיחק – המחושב לפי כמות המשחקים שהקבוצה שיחקה + כמות המשחקים שהשחקן שיחק ללא הקבוצה. כדי לא להוסיף אליו משחקים לא רלוונטיים (שהקבוצה שיחקה לפני שהשחקן הצטרף), נחסיר מכמות המשחקים של השחקן את כמות המשחקים שהקבוצה שיחקה לפני הצטרפותו.
נכניס אותו לשני עצי AVL הכלליים של השחקנים: אחד המסודר לפי מזהה השחקן, והשני מסודר לפי ה-score של השחקן. נעדכן את המצביע של השחקן עם ה-score הגבוה ביותר, ונעדכן את המצביעים לשחקנים "הקרובים" ביותר מבחינת score, ובכן נעדכן בהתאם גם את המצביעים של השחקנים ה"קרובים".
נוסיף את השחקן לקבוצה מתאימה: נוסיף את השחקן לשני עצי AVL של שחקני הקבוצה: עץ אחד המסודר לפי מזהה הקבוצה, ועץ שני המסודר לפי ה-score של שחקני הקבוצה. בנוסף לכך, נעדכן את פרטי הקבוצה בהתאם לנתונים של השחקן החדש: נוסיף את כמות השערים של השחקן לכמות סה"כ שהבקיעה הקבוצה, ונוסיף את כמות הכרטיסים של השחקן לכמות סה"כ כרטיסים שקיבלה הקבוצה. כמו כן, נוסיף 1 לכמות השוערים של הקבוצה אם השחקן החדש יכול לשחק כשוער. נעדכן את המצביע של השחקן בקבוצה עם ה-score הגבוה ביותר. לאחר מכן, נוסיף את הקבוצה לm_qualifiedTeams במקרה שכעת הקבוצה בעלת לפחות 11 שחקנים ויש לפחות שוער אחד.
אם זהו אכן המקרה, נעדכן את המצביעים של אותה קבוצה לקבוצות הסמוכות אליו מבחינת מזהי הקבוצה, ונעדכן את המצביעים של הקבוצות הסמוכות. לאורך כל הפעולה, במקרה של כישלון בהקצאת הזיכרון, נחזיר ALLOCATION_ERROR. בסוף, נוסיף 1 לכמות השחקנים שיש סה"כ בגביע העולם, ונחזיר SUCCESS.

<u>סיבוכיות זמן:</u>
בפעולה זו, מחפשים אם השחקן בעץ כל השחקנים בסיבוכיות O(log(n)). לאחר מכן, נכניס אותו לשני העצים בסיבוכיות O(log(n)*2) (הכנסה + איזון העץ) לכל עץ כאשר n היא כמות השחקנים סה"כ בגביע העולם. כמו כן, מחפשים בעץ כל הקבוצות עבור הקבוצה הנדרשת בסיבוכיות של O(log(k)) כאשר k היא כמות הקבוצות סה"כ בגביע העולם. בקבוצה הספציפית של השחקן, מכניסים את השחקן לשני העצים של שחקני הקבוצה. בכל אחד מהעצים יש מקסימום n שחקנים, ולכן סיבוכיות החיפוש וההכנסה של כל אחד מהעצים הוא O(log(n)*2). מאזנים את כל אחד מהעצים הללו בעזרת גלגולים בשיטה שנלמדה בהרצאה, בסיבוכיות O(1).
גם מתעדכן המצביע לשחקן שהבקיע הכי הרבה שערים – גם של המשחק כולו וגם של הקבוצה. פעולה זו דורשת חיפוש בעצים של השחקנים עם מקסימום n שחקנים ובכן בסיבוכיות של O(log(n)).
בנוסף, מעדכנים את המצביעים לשחקנים הכי "קרובים" מבחינת score –כל עדכון דורש 2 חיפושים בעץ השחקנים בסיבוכיות O(log(n)). מעדכנים זאת עבור שלוש שחקנים שונים (השחקן החדש ושני השחקנים שסמוכים אליו), ולכן סך הסיבוכיות הוא O(log(n)*6).
יש אפשרות להכנסת הקבוצה לעץ הקבוצות הכשרות, ולכן זוהי בסיבוכיות O(log(k)*2 (עבור הכנסה + איזון העץ).
בהתאם לכך, מעדכנים את המצביעים לקבוצות בעלות מזהים סמוכים. גם כאן, כל עדכון דורש שני חיפושים בעץ הקבוצות ולכן בעלות סיבוכיות O(log(k)). מעדכנים 3 קבוצות (הקבוצה הנוכחית ושתי הקבוצות הסמוכות) ולכן סך הסיבוכיות הוא O(log(k)*6). סך הכל סיבוכיות הזמן הוא: **O(log(n)+log(k))** = O(log(n)*16 + log(k)*9).

- remove_player

בפעולה זו, נסיר את השחקן מגביע העולם כולו.
נבדוק את תקינות הקלט, ונחזיר INVALID_ID אם התקבל מזהה שחקן קטן או שווה ל-0. נחפש את השחקן בעץ כל השחקנים המסודר לפי מזהה שחקן, ונחזיר FAILURE אם השחקן לא קיים. אחרת, ניגש לקבוצה בה משחק השחקן דרך המצביע שנמצא באובייקט השחקן, ונסיר אותו מהקבוצה על ידי הסרתו משני העצים של השחקנים של הקבוצה: אחד המסודר לפי מזהה שחקן, והשני לפי ה-score שלו. בנוסף לכך, נעדכן את נתוני הקבוצה בהתאם: נחזיר את כמות השערים, הכרטיסים, סך השחקנים וכמות השוערים של הקבוצה. אם כעת הקבוצה כבר אינה

כשרה למשחק (מכילה פחות מ-11 שחקנים או שאין בה שוערים) נסיר את הקבוצה מהעץ: m_qualifiedTeams, ונעדכן את המצביעים של הקבוצה לקבוצות עם מזהים סמוכים, וגם נעדכן את המצביעים של הקבוצות הסמוכות.
נסיר את השחקן משני העצים של שחקני גביע העולם: אחד המסודר לפי מזהה שחקן והשני לפי ה-score שלו.
נעדכן את המצביעים לשחקן עם ה-score הכי הגבוה, גם של כלל גביע העולם וגם של הקבוצה בה השחקן שיחק.
נעדכן את המצביעים של השחקנים שהיו הכי "קרובים" לשחקן מבחינת score, ונשחרר את הזיכרון של השחקן.
נחסיר 1 מספירת כלל שחקני גביע העולם, ובסוף נחזיר SUCCESS.

<u>סיבוכיות הזמן:</u>
בפעולה זו, אנו מחפשים את השחקן בעץ כל השחקנים בסיבוכיות O(log(n)) כאשר n היא סך כל השחקנים במערכת. בנוסף לכך, אנו מסירים את השחקן מארבע עצים שונים: שני העצים של שחקנים המסודרים לפי score: אחד של כלל גביע העולם, השני של הקבוצה של השחקן ושני העצים של שחקנים המסודרים לפי מזהה שחקן.
בכל אחד מהעצים יש לכל היותר n שחקנים, ולכן סיבוכיות ההסרה היא O(log(n)*2) לכל עץ, שכולל חיפוש בעצים ואז גלגולים נדרשים כדי לאזן את העצים. עדכון המצביעים של השחקנים עם ה-score הגבוה ביותר הוא גם בסיבוכיות O(log(n)) לכל מצביע (אחד של כלל גביע העולם, השני של הקבוצה הספציפית).
אנו גם מסירים את הקבוצה מעץ הקבוצות הכשרות m_qualifiedTeams, אשר היא מכילה רק את הקבוצות עבורן יש לפחות 11 שחקנים – נסמן מספר זה כ-m. לכן בהכרח כמות הקבוצות (nodes) בעץ הזה הוא קטן <u>ממש</u> מכמות השחקנים במערכת: m < n. הסרת הקבוצה נעשה בסיבוכיות O(log(m)*2) סך הכל, אשר כולל גם גלגולים שנדרשים כדי לאזן את העץ. לכן סך פעולות אלו הן בסיבוכיות זמן של O(log(m)*2) = O(log(m)) = O(log(n)).
עדכון נתוני הקבוצה וגביע העולם נעשה באופן ישיר למשתנים אשר הם members של Team ושל world_cup_t ולכן מוסיפים סיבוכיות זמן של O(1). כמו כן, עדכון השחקנים שהכי קרובים לשחקן מבחינת score הוא בסיבוכיות O(1) משום שהעדכון הוא ישיר דרך המצביעים שהחזיק השחקן. באופן דומה, עדכון המצביעים של הקבוצה לקבוצות הסמוכות מבחינת מזהי הקבוצה הוא גם בסיבוכיות O(1).
לכן סך סיבוכיות הזמן הוא: O(log(n)*10) = **O(log(n))**.

– update_player_stats

בפעולה זו, נעדכן את נתוני השחקן.
ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה שחקן קטן או שווה ל-0, או שהתקבל נתון אחר שלילי. נחפש את השחקן בעץ כל השחקנים המסודר לפי מזהה שחקן, ונחזיר FAILURE אם השחקן לא קיים.
אחרת, ניגש דרך השחקן לקבוצה שבה משחק, באמצעות מצביע לקבוצה אשר קיים באובייקט השחקן. נסיר את השחקן משני העצים של שחקנים המסודרים לפי score של השחקנים: אחד של כלל גביע העולם, והשני של הקבוצה בה השחקן משחק. נעדכן את השחקן לפי הנתונים מהקלט, ואז נחזיר חזרה את השחקן המעודכן לתוך שני העצים של שחקנים המסודרים לפי score של השחקנים: אחד של כלל גביע העולם, והשני של הקבוצה בה השחקן משחק. נעדכן את המצביעים לשחקן עם ה-score הכי גבוה בהתאם לכך. לאחר מכן, נעדכן את נתוני הקבוצה: נוסיף את השערים לסך כל השערים שהבקיעו כל שחקני הקבוצה, ונוסיף את כמות הקלפים לסך כל הקלפים של הקבוצה. בנוסף, נעדכן את המצביעים לשחקנים "הקרובים" ביותר לשחקן מבחינת score, ובכן נעדכן גם את המצביעים של הקבוצות האלו.
לאורך כל הפעולה, במקרה של כישלון בהקצאת זיכרון נחזיר ALLOCATION_ERROR. בסוף נחזיר SUCCESS.

<u>סיבוכיות הזמן:</u>
בפעולה זו, אנו מחפשים את השחקן בעץ כל השחקנים בסיבוכיות O(log(n)) כאשר n היא סך כל השחקנים במערכת. בנוסף לכך, אנו מסירים ואז מכניסים את השחקן לשני העצים של שחקנים המסודר לפי score: אחד של כלל גביע העולם, השני של הקבוצה של השחקן. בכל אחד מהעצים יש לכל היותר n שחקנים, ולכן סיבוכיות ההכנסה וההסרה היא O(log(n)*2) לכל עץ, שכולל חיפוש בעצים ואז גלגולים נדרשים כדי לאזן את העצים. עדכון המצביעים של השחקנים עם ה-score הגבוה ביותר הוא בסיבוכיות O(log(n)) לכל מצביע. עדכון המצביעים לשחקנים ה"קרובים" מבחינת score הוא בסיבוכיות O(log(n)*2) לכל שחקן, כאשר אנו מעדכנים 5 שחקנים סך הכל (השחקן הנוכחי, שני השחקנים שהיה סמוך אליהם לפני העדכון והשניים שכעת סמוך אליהם אחרי העדכון).
עדכון נתוני השחקן והקבוצה נעשה באופן ישיר למשתנים אשר הם members של Player ו- Team ולכן מוסיפים סיבוכיות זמן של O(1). לכן סך סיבוכיות הזמן הוא: O(log(n)*17) = **O(log(n))**.

– play_match

בפעולה זו, שני שחקנים משחקים אחד מול השני.
ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם אחד ממזהי הקבוצות קטנים או שווים ל-0, או שהתקבלו מזהי קבוצות זהות. נחפש את שתי הקבוצות הללו בעץ m_qualifiedTeams, אשר מכיל מלכתחילה את הקבוצות שכשרות למשחק – עם לפחות 11 שחקנים ושוער אחד. במקרה שלפחות אחת מהקבוצות לא כשרה או לא קיימת, נחזיר FAILURE. הקבוצה המנצחת בעלת הסכום הגבוה ביותר של: כמות הנקודות + כמות השערים של סך שחקניה – כמות הקלפים של סך שחקניה. הקבוצה המנצחת תזכה בעוד 3 נקודות, ובמקרה של תיקו, שתי הקבוצות יזכו בנקודה אחת. בסוף נחזיר SUCCESS.

<u>סיבוכיות הזמן:</u>
בפעולה זו, אנו מחפשים את שתי הקבוצות בעץ m_qualifiedTeams, אשר מכיל מקסימום k קבוצות (מתוך סך k הקבוצות בכלל גביע העולם). לכן החיפוש היא בסיבוכיות O(log(k)*2) = O(log(k)). הגישה לכל אחד מהנתונים של הקבוצות, כמו סך השערים שכל שחקניה הבקיעו היא ישירה דרך משתנים שהם members של מחלקת Team. ולכן בסיבוכיות O(1) לכל משתנה. לכן סך סיבוכיות הזמן הוא **O(log(k))**.

– get_num_played_games

בפעולה זו, נחזיר את כמות המשחקים שהשחקן שיחק סה"כ.
ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה שחקן שלילי. לאחר מכן, נחפש את השחקן בעץ כל השחקנים לפי מזהה שחקן (m_playersByID). אם לא קיים שחקן כזה, נחזיר FAILURE. אחרת, ניגש דרך השחקן אל הקבוצה שבה הוא משחק (יש באובייקט השחקן מצביע ישירות לקבוצה). נשלוף מהקבוצה את כמות המשחקים שהקבוצה שיחקה סך הכל, ונוסיף את ערך זה לכמות המשחקים שהשחקן שיחק. נחזיר את הסכום שהתקבל.

<u>סיבוכיות הזמן:</u>
בפעולה זו, אנו מחפשים בעץ כל השחקנים בשיטה שנלמדה בהרצאה בסיבוכיות O(log(n)) כאשר n הוא כמות השחקנים בגביע העולם. בנוסף לכך, אנו ניגשים ישירות לקבוצה בה השחקן משחק דרך המצביע בשחקן – בסיבוכיות O(1). לכן סך הכל, סיבוכיות הזמן הוא **O(log(n))**.

– get_team_points

בפעולה זו, נחזיר את כמות הנקודות שהקבוצה צברה סה"כ. ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה קבוצה שלילי. לאחר מכן, נחפש את הקבוצה בעץ כל הקבוצות לפי מזהה קבוצה (m_teamsByID). אם לא קיימת קבוצה כזו, נחזיר FAILURE. אחרת, נחזיר את כמות הנקודות של אותה קבוצה.

<u>סיבוכיות הזמן:</u>
בפעולה זו, אנו מחפשים בעץ כל הקבוצות בשיטה שנלמדה בהרצאה בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בגביע העולם. בקבוצה, אנו ניגשים ישירות למשתנה מטיפוס int אשר הוא member של מחלקת Team, ולכן בסיבוכיות הזמן של O(1). לכן סך הכל, סיבוכיות הזמן הוא **O(log(k))**.

– unite_teams

בפעולה זו, נאחד בין שתי קבוצות הנתונות לפי מספר ID אל תוך קבוצה חדשה.
ראשית, נבדוק את תקינות הקלט שקיבלנו - נבדוק שהמספרי ID תקינים לפי הגדרת מספרי ID. לאחר מכן, נחפש בעץ של הקבוצות (m_teamsyID) לבדוק האם קיימת כבר קבוצה עם הID החדש שנתון לנו. אם קיימת קבוצה כזו, והיא לא אחת הקבוצות המתאחדות, פעולה זו לא חוקית ולכן נחזיר FAILURE. לאחר מכן, נבדוק שאכן שתי הקבוצות המתאחדות קיימות במערכת, על ידי חיפוש בעץ m_teamsByID AVL. אם אחת הקבוצות לא נמצאות, נחזיר FAILURE. אחרי בדיקת הקלט, נקצה זיכרון ונאתחל TEAM חדש עם המזהה הנתון. נקרא לפונקציית עזר unite_teams המאחדת את שתי הקבוצות על ידי הפעולות הבאות:

- חישוב הנתונים היבשים של הקבוצה החדשה על פי סכום הנתונים של שתי הקבוצות המתאחדות (numCards, numGames, etc.).

- איחוד עצי השחקנים של שתי הקבוצות והכנסה לעץ של הקבוצה המאוחדת (פירוט בסיבוכיות זמן).
- עדכון כמות המשחקנים שכל שחקן שיחק, כדי לאתחל מחדש את ספירת המחשקים בקבוצה החדשה.
- עדכון הtopScorer על ידי מעבר על העץ m_playersByScore החדש (search_and_return_max).

לאחר איחוד שתי הקבוצות, נעדכן את הקבוצה שכל שחקן מצביע אליו, לקבוצה החדשה.
נסיר את שתי הקבוצות הישנות מהעץ m_qualifiedTeams (אם הן לא נמצאות, נתפוס את החריגה שנזרקה ונתקדם), ומהעץ m_teamsByID. נכניס את הקבוצה החדשה שבנינו אל תוך העץ m_teamsByID על ידי פעולת insert של העץ AVL. נבדוק האם הקבוצה החדשה חוקית בשביל לשחק, ואם כן, נוסיף אותה לעץ m_qualifiedTeams. נעדכן את המצביעים לקבוצות הסמוכות, של הקבוצות שהיו סמוכות שתי הקבוצות שאיחדנו, ובכן נעדכן המצביעים לקבוצות הסמוכות של הקבוצה החדשה. לאורך כל הפעולה, במקרה של כישלון בהקצאת זיכרון נחזיר ALLOCATION_ERROR. נשחרר את הזכרון של שתי הקבוצות הקודמות ובסוף נחזיר SUCCESS.

<u>סיבוכיות הזמן:</u>
בפעולה זו, אנחנו מתחילים מ3 חיפושים בתוך העץ m_teamsByID, הנעשו בשיטה הנלמדה בהרצאה, בסיבוכיות זמן של O(log(k)), כאשר k הוא מספר הקבוצות במערכת. באיחוד של הקבוצות, גישה לנתונים של שתי הקבוצות הישנות וסכומם נעשה ב$O(1)$. נסמן ב$n_{TEAMID1}$ ו-$n_{TEAMID2}$ את כמות השחקנים בשתי הקבוצות בהתאמה.
נמזג בין עצי השחקנים של שתי הקבוצות בסיבוכיות $O(n_{TEAMID1}+ n_{TEAMID2})$ בשיטה שנלמדה בתרגול:

- השמת עצי השחקנים של שתי הקבוצות במערכים לפי סדר Inorder – $O(n_{TEAMID1}+ n_{TEAMID2})$
- איחוד המערכים של m_playersByID של שתי הקבוצות תוך שמירה על סדר עולה, ואז איחוד המערכים של m_playersByScore של שתי הקבוצות תוך שמירה על סדר עולה – $O(n_{TEAMID1}+ n_{TEAMID2})$
- בניית עץ לפי המערכים לפי הסדר בשיטת Inorder Insert (<u>לא</u> ההכנסה הרגילה) – $O(n_{TEAMID1}+ n_{TEAMID2})$

פעולת update_team_id דורש מעבר על כל שחקן בעץ באופן רקורסיבי, ולכן בעל סיבוכיות זמן של $O(n_{TEAMID1}+ n_{TEAMID2})$. מסירים את הקבוצות הישנות משתי עצי הקבוצות בסיבוכיות של O(log(k)*2) (הוצאה + איזון העץ עם גלגולים). בשתי ההוצאות מהעץ m_qualifiedTeams, יש צורך בעדכון המצביעים לקבוצות הסמוכמות כדי שידלגו על הקבוצה שהוסרה ולכן בסיבוכיות $O(1)$. כמו כן, פעולת הinsert הנעשה במקסימום פעמיים לוקח זמן של O(log(k)*2) (הוצאה + איזון העץ עם גלגולים). עדכון המצביעים לקבוצות הסמוכות לקבוצה החדשה הוא בסיבוכיות של O(log(k)*2) לכל עדכון, ואנו מעדכנים סך הכל שלוש קבוצות.
לכן בסך הכל, נקבל סיבוכיות זמן: $O(log(k)*21+(n_{TEAMID1}+n_{TEAMID2})*2)$ = **$O(log(k)+n_{TEAMID1}+n_{TEAMID2})$**.

– get_top_scorer

בפעולה זו, נחזיר את מזהה השחקן עם ה-score הכי גבוה. ה-score של השחקן נקבע לפי הפרמטרים הבאים: השחקן עם כמות השערים הגבוה ביותר ששחקן הבקיע. במקרה של שוויון, אז השחקן עם כמות הקלפים הנמוך ביותר. במקרה של שוויון, אז השחקן עם מזהה הקבוצה הכי גבוה.
ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה קבוצה השווה ל-0. כמו כן, אם אין שחקנים כלל המשתתפים בגביע העולם, נחזיר FAILURE.

- אם התקבל מזהה קבוצה הגדולה מ-0, אז נחפש את הקבוצה בעץ כל הקבוצות לפי מזהה קבוצה (m_teamsByID). אם לא קיימת קבוצה כזו או אם בקבוצה זו אין שחקנים אז נחזיר FAILURE. אחרת, נחזיר את מזהה השחקן שמשחק בקבוצה זו עם ה-score הכי גבוה מתוך כלל השחקנים בקבוצה זו.
- אחרת, התקבל מזהה קבוצה הקטנה מ-0, ולכן נחזיר את מזהה השחקן עם ה-score הכי גבוה מתוך כלל השחקנים בגביע העולם.

<u>סיבוכיות הזמן:</u>
בפעולה זו, במקרה שמזהה הקבוצה גדולה מ-0, אז מחפשים בעץ כל הקבוצות בשיטה שנלמדה בהרצאה בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בגביע העולם. בקבוצה, ניגשים ישירות למצביע של השחקן עם ה-score הגבוה ביותר בין השחקנים באותה קבוצה, אשר הוא member של מחלקת Team, ולכן בסיבוכיות הזמן של $O(1)$. לכן סך הכל, סיבוכיות הזמן הוא **O(log(k))**.
במקרה שמזהה הקבוצה קטנה מ-0, אז ניגש ישירות למצביע לשחקן עם ה-score הגבוה ביותר בגביע העולם, אשר הוא member של מחלקת world_cup_t. לכן סיבוכיות הזמן סך הכל הוא **O(1)**.

מבני נתונים 1 – 234218
שרה גריפית – ת"ז: 341312304
ליאור בר יוסף – ת"ז: 207022443

- get_all_players_count

בפעולה זו, נחזיר את מספר השחקנים בקבוצה או בגביע העולם.
ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה קבוצה השווה ל-0.

- אם התקבל מזהה קבוצה הגדול מ-0, אז נחפש את הקבוצה בעץ כל הקבוצות לפי מזהה קבוצה (m_teamsByID). אם לא קיימת קבוצה כזו נחזיר FAILURE. אחרת, נחזיר כמות השחקנים בקבוצה.
- אחרת, התקבל מזהה קבוצה הקטנה מ-0, ולכן נחזיר את כמות השחקנים בגביע העולם.

<u>סיבוכיות הזמן:</u>
בפעולה זו, במקרה שמזהה הקבוצה גדולה מ-0, אז מחפשים בעץ כל הקבוצות בשיטה שנלמדה בהרצאה בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בגביע העולם. בקבוצה, ניגשים ישר למשתנה מטיפוס int שמחזיק את כמות השחקנים באותה קבוצה, אשר הוא member של מחלקת Team, ולכן בסיבוכיות הזמן של O(1).
לכן סך הכל, סיבוכיות הזמן הוא **O(log(k))**.
במקרה שמזהה הקבוצה קטן מ-0, אז ניגש ישירות למשתנה מטיפוס int שמחזיק את כמות השחקנים בגביע העולם, אשר הוא member של מחלקת world_cup_t. לכן סיבוכיות הזמן סך הכל הוא **O(1)**.

– get_all_players

בפעולה זו, נחזיר את מזהי כל השחקנים בקבוצה או בגביע העולם.
ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה קבוצה השווה ל-0 או שהמערך שהתקבל מצביע ל-NULL. נחזיר FAILURE אם לא קיימים כלל שחקנים בגביע העולם.

- אם התקבל מזהה קבוצה הגדול מ-0, אז נחפש את הקבוצה בעץ כל הקבוצות לפי מזהה קבוצה (m_teamsByID). אם לא קיימת קבוצה כזו או שאין בה שחקנים, אז נחזיר FAILURE. אחרת, נעבור על כל השחקנים בקבוצה <u>בסדר עולה</u> לפי ה-score של השחקנים (בעצם נעבור על העץ m_playersByScore של אותה קבוצה בשיטת in-order), ונכניס את מזהה השחקנים למערך שהתקבל בקלט.
- אחרת, התקבל מזהה קבוצה הקטנה מ-0, ונעבור על כל השחקנים בגביע העולם בסדר עולה לפי ה-score של השחקנים (בעצם נעבור על העץ m_playersByScore של כל גביע העולם בשיטת in-order), ונכניס את מזהה השחקנים למערך שהתקבל בקלט.

בסוף נחזיר SUCCESS.

<u>סיבוכיות הזמן:</u>
בפעולה זו, במקרה שמזהה הקבוצה גדול מ-0, אז מחפשים בעץ כל הקבוצות בשיטה שנלמדה בהרצאה בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בגביע העולם. בקבוצה, עוברים על כלל השחקנים בקבוצה, אשר נסמן כ-n_{TEAMID}. לכן סך הסיבוכיות זמן במקרה זה הוא **$O(\log(k)+n_{TEAMID})$**.
במקרה שמזהה הקבוצה קטנה מ-0, אז נעבור על כלל השחקנים בכל גביע העולם, אשר נסמן ב-n. לכן סיבוכיות הזמן עבור מקרה זה הוא **O(n).**

- get_closest_player

בפעולה זו, נחזיר את מזהה השחקן שהכי "קרוב" לשחקן עם מזהה מהקלט מבחינת score.
"קירבה" ב-score מוגדר באופן הבא: השחקן הקרוב הוא זה עם כמות ה-goals הכי קרוב אליו. במקרה של שוויון, אז השחקן הקרוב הוא זה עם כמות השערים הכי קרוב אליו. במקרה של שוויון, אז השחקן הקרוב הוא זה עם מזהה השחקן הכי קרוב אליו. במקרה של שוויון, אז השחקן הקרוב הוא זה עם מזהה השחקן הגבוה ביותר.
ראשית נבדוק את הקלט, ונחזיר INVALID_INPUT אם התקבל מזהה קבוצה או מזהה שחקן הקטן או שווה ל-0.
נחזיר FAILURE אם קיימים פחות משני שחקנים בכלל גביע העולם, או אם לא קיימת קבוצה עם המזהה הנדרש.
בנוסף, נחזיר FAILURE אם קיימת קבוצה כזו, אך לא קיים לה שחקן עם המזהה הנדרש. נחפש על עץ השחקנים לפי מזהה השחקן, ונשווה בין השחקים שהכי "קרובים" אליו באופן הבא: באותו שחקן יש שני מצביעים לשחקנים שהכי "קרובים" אליו – אחד עם score שגדול ממנו, והשני עם score שקטן ממנו. דרכם ניגש ישירות לשני השחקנים הללו ונשווה למי יש את ההפרש הקטן ביותר בנתונים עם השחקן הדרוש.
בסוף נחזיר את מזהה השחקן שהכי "קרוב" אליו מבחינת score.

מבני נתונים 1 – 234218
שרה גריפית – ת"ז: 341312304
ליאור בר יוסף – ת"ז: 207022443

<u>סיבוכיות הזמן:</u>
בפעולה זו, אנו מחפשים בעץ כל הקבוצות המסודר לפי מזהה קבוצה את הקבוצה הנדרשת, בשיטה שנלמדה בכיתה בסיבוכיות O(log(k)) כאשר k היא כמות הקבוצות בגביע העולם. לאחר מכן, אנו מחפשים בעץ כל השחקנים של אותה קבוצה את השחקן הדרוש. אם n_{TEAMID} היא כמות השחקנים בקבוצה, אז סיבוכיות הזמן של החיפוש הוא $O(\log(n_{TEAMID}))$. לאחר מכן, אנו ניגשים ישירות דרך השחקן שמצאנו לשני השחקנים שהכי קרובים אליו. לכן הגישה אליהם ובדיקת מי הכי "קרוב" לשחקן הדרוש מוסיף סיבוכיות זמן של O(1).
לכן סך הכל, נקבל סיבוכיות זמן של **$O(\log(k)+\log(n_{TEAMID}))$.**

- knockout_winner

בפעולה זו, אנחנו מבצעים תחרות פנימית בין כמות קבוצות מסויימת, ונחזיר את מזהה הקבוצה של הקבוצה המנצחת. קודם כל, נבדוק את תקינות הקלט, ונחזיר INVALID_INPUT במקרה שאינו תקין. אחרת, נספור את כמות הקבוצות הכשרות לשחק בטווח המזהים הנתון. אם לא קיימת קבוצה כזו, נחזיר FAILURE.
נספור את הקבוצות בעץ AVL של m_qualifiedTeams (המכילה רק את הקבוצות הכשרות - בעלות לפחות 11 שחקנים ושוער אחד), ונקצה זיכרון למערך בגודל של כמות הקבוצות הכשרות בטווח הנתון. נמלא את המערך בעותק של הקבוצות המתאימות (העותק <u>אינו</u> כולל את שחקני הקבוצות כלל, אלא רק את הנתונים של הקבוצה (משתנים של int) וגם המצביעים <u>לעותקים</u> של הקבוצות הסמוכות אליו מימין ומשמאל).
נעבור על כל הקבוצות הכשרות החל מהקבוצה המינימלית עד המקסימלית, בעזרת המצביעים שיש לכל קבוצה לקבוצות הסמוכות אליו, ונבצע רצף משחקים ביניהם. כל זוג קבוצות ישחקו משחק, כאשר הקבוצה המפסידה מתאחדת עם המנצחת, ומעדכנים את המצביעים של הקבוצות הסמוכות כדי לדלג על הקבוצה שהפסידה. חוזרים על תהליך זה עד שנשארת קבוצה בודדת במערך אשר היא המנצחת בטורניר. חשוב להדגיש כי כל הטורניר מתרחש בין עותקים של הקבוצות, ולכן השינויים שנעשים במהלך פעולה זו <u>לא</u> משפיעים לתווך ארוך על הקבוצות עצמן. לאורך כל הפעולה, במקרה של כישלון בהקצאת זיכרון נחזיר ALLOCATION_ERROR. בסוף נחזיר את מזהה השחקן שניצח בטורניר.

<u>סיבוכיות זמן:</u>
ראשית, נספור את כמות הקבוצות הכשרות שיש בטווח הנתון. נחפש ראשית על העץ m_qualifiedTeams את השחקן עם מזהה הקבוצה המינימלי בטווח הנתון. במהלך החיפוש במקרה הגרוע, יורדים למטה ואז למעלה את כל העץ. הקבוצות בעץ זה בעלי לפחות 11 שחקנים, ולכן אם נסמן את כמות הקבוצות בעץ זה כ-m וכמות כלל השחקנים כ-n, אז m<n. אך הוא גם קטן או שווה לסך כל הקבוצות בגביע העולם, שנסמן כ-k, ולכן m≤k. לכן החיפוש עבור הקבוצה מינימלית נעשית בסיבוכיות זמן של O(log(m)*2) = O(log(m)), כאשר $m \le k,\ m < n$. נעבור על כל הקבוצות הכשרות החל מהקבוצה המינימלית בעזרת המצביעים של הקבוצות הסמוכות. יש מקסימום r קבוצות, ולכן סיבוכיות הזמן של מעבר, ספירת והעתקת הקבוצות הוא O(r). עדכון המצביעים של הקבוצות הסמוכות נעשה במעבר על כלל הקבוצות ולכן בסיבוכיות של O(r). חישוב המנצחים בכל משחק הוא בעזרת גישה ישירה למשתנים של הקבוצות לכן סיבוכיות הזמן עבור כל משחק הוא O(1). עדכון המצביעים לקבוצות הסמוכות בכל משחק (כדי לדלג על הקבוצה שהפסידה) נעשה באופן ישיר בעזרת המצביעים ולכן זהו בסיבוכיות O(1).
כמות הקבוצות שמשחקות בשלב הראשון הוא r (כל הקבוצות בטווח). בשלב השני של הטורניר, כבר חצי מהקבוצות הצטמצמו, והמעבר יהיה רק על מחצית הקבוצות (בעזרת המצביעים שעודכנו להצביע על הקבוצות שעדיין משחקות בטורניר ולא הפסידו). בשלב השלישי, יש מחצית מהקבוצות שהיו בשלב השני. וכן הלאה...
לכן סך הכל, בטורניר נעבור על 2r קבוצות:

$$r+\frac{r}{2}+\frac{r}{4}+\frac{r}{8}\ldots = r+r\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{8}\ldots\right) = r+r*(\sum_{n=1}^{\infty}\left(\frac{1}{2}\right)^n = {}_{\text{סכום טור הנדסי}}\ r+r*(1) = 2r$$

לכן סך סיבוכיות הזמן של כלל משחקי הטורניר הוא O(2r)=O(r).
לכן, בסך הכל, פונקציה זו לוקחת זמן של: O(log(m)+r*2+r*2) = O(log(m)+r)
נזכר כי m<n וגם m≤k, לכן: $m \le min\{k,n\}$
לכן בסך הכל, נקבל כי פונקציה זו לוקחת זמן של: **O(log(min{k,n}) + r)**.