מתרגל ממונה על התרגיל: אורי קורן, uri.koren@campus.technion.ac.il

תאריך ושעת הגשה: 28.1 בשעה 23:59

אופן ההגשה: בזוגות. אין להגיש ביחידים. (אלא באישור מתרגל אחראי של הקורס)

הנחיות כלליות:

- שאלות על התרגיל יש לפרסם באתר הפיאצה של הקורס תחת לשונית "hw-wet-2":
  - האתר: https://piazza.com/class/m34xgygziyn64l, נא לקרוא את השאלות של סטודנטים אחרים לפני שמפרסמים שאלה חדשה, למקרה שנשאלה כבר.
- נא לקרוא את המסמך "נהלי הקורס" באתר הקורס. בנוסף, נא לקרוא בעיון את כל ההנחיות בסוף מסמך זה.
- בפורום הפיאצה ינוהל FAQ ובמידת הצורך יועלו תיקונים כ**הודעות נעוצות** (Pinned Notes). תיקונים אלו **מחייבים**.
- התרגיל מורכב משני חלקים: יבש ורטוב.
  - לאחר קריאת כלל הדרישות, מומלץ לתכנן תחילה את מבני הנתונים על נייר. דבר זה יכול לחסוך לכם זמן רב.
  - לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסיבוכיות בתרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
  - את הפתרון שלכם מומלץ לחלק למחלקות שונות שאפשר לממש (ולבדוק!) בהדרגתיות.
  - המלצות לפתרון התרגיל נמצאות באתר הקורס תחת: "Programming Tips Session".
    - המלצות לתכנות במסמך זה אינן מחייבות, אך מומלץ להיעזר בהן.
  - חומר התרגיל הינו כל החומר שנלמד בהרצאות ובתרגולים עד וכולל UnionFind.
- העתקת תרגילי בית רטובים תיבדק באמצעות תוכנת בדיקות אוטומטית, המזהה דמיון בין כל העבודות הקיימות במערכת, גם כאלו משנים קודמות. לא ניתן לערער על החלטת התוכנה. התוכנה אינה מבדילה בין מקור להעתק! אנא הימנעו מהסתכלות בקוד שאינו שלכם.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת:
  jonathan.gal@campus.technion.ac.il

הקדמה:

לאחר שעזרתם להנדסת חומרים לעקוב אחרי סוסי הפרא הרבים, הסטודנטים של הפקולטה החליטו להקים ליגת רכיבה על סוסים. עלינו לעזור לסטודנטים של הפקולטה להקים מערכת יעילה שתעקוב אחרי קבוצות הרוכבים השונות ומאזן הניצחונות שלהן. מאזן של רוכב הוא סה"כ הניצחונות שלו פחות סה"כ ההפסדים שלו. המאזן של קבוצה זה סה"כ הניצחונות של הרוכבים בקבוצה פחות סה"כ ההפסדים של הרוכבים בקבוצה.

סימונים לצורכי סיבוכיות:

נסמן ב-n את מספר רוכבי הסוסים במערכת.

ב-m את מספר קבוצות הרוכבים שנוספו בהצלחה למערכת (כפי שמפורט בהמשך, כולל גם את הקבוצות שנמחקו כתוצאה מאיחוד מוצלח).

דרוש מבנה נתונים למימוש הפעולות הבאות:

plains_t()

מאתחלת מבנה נתונים ריק. תחילה אין במערכת עדרים או סוסים.

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

---

virtual ~plains_t()

הפעולה משחררת את המבנה (כל הזיכרון אותו הקצאתם חייב להיות משוחרר).

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(n+m)$ במקרה הגרוע.

StatusType add_team(int teamId)

הפעולה מוסיפה למבנה נתונים קבוצה חדשה ללא רוכבים.

פרמטרים:

teamId	מזהה הקבוצה שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId <=0.
FAILURE	אם בעבר כבר נוספה בהצלחה קבוצה עם מזהה teamId למבני הנתונים.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ בממוצע על הקלט משוערך עם עצמו.

StatusType add_jockey(int jockeyId, int teamId)

רוכב בעל מזהה ייחודי jockeyId מצטרף לקבוצה teamId. הרוכב מתחיל עם מאזן ניצחונות אפס.

פרמטרים:

jockeyId	מזהה הרוכב שצריך להוסיף.
teamId	מזהה הקבוצה של הרוכב.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם jockeyId <=0 או אם teamId <=0.
FAILURE	אם קיים כבר רוכב במזהה jockeyId או שהקבוצה במזהה teamId לא קיימת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ בממוצע על הקלט משוערך עם עצמו.

StatusType update_match(int victoriousJockeyId, int losingJockeyId)

לאחר משחק בין רוכבים משתי קבוצות שונות, רוצים לעדכן שרוכב במזהה victoriousJockeyId ניצח את losingJockeyId. לאחר פקודה מוצלחת, על המאזן של הרוכב במזהה victoriousJockeyId לגדול ב-1, ועל המאזן של הרוכב במזהה losingJockeyId לקטון ב-1 (מאזן יכול להיות שלילי).

פרמטרים:

victoriousJockeyId	מזהה הרוכב המנצח.
losingJockeyId	מזהה הרוכב המפסיד.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם losingJockeyId <=0, victoriousJockeyId <= 0, losingJockeyId==victoriousJockeyID
FAILURE	אם אין רוכבים עם המזהים losingJockeyId או victoriousJockeyId, או ששני הרוכבים נמצאים באותה קבוצה.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* m)$ בממוצע על הקלט משוערך יחד עם merge_teams ו-unite_by_record.

StatusType merge_teams(int teamId1, int teamId2)

הקבוצות בעלות המזהים teamId1 ו-teamdId2 מתאחדות (כלומר לאחר האיחוד כל הרוכבים משתי הקבוצות נמצאים בקבוצה משותפת), והמזהה החדש שלהן הוא הקבוצה עם המאזן ניצחונות הטוב יותר (במקרה של שוויון המזהה החדש הוא teamdId1). לאחר האיחוד, אם בה"כ מזהה הקבוצה החדש הוא teamId1, אז לא קיימת יותר קבוצה במזהה teamId2, <u>וגם לא ניתן להוסיף קבוצה חדשה במזהה זה</u>.

<u>פרמטרים</u>:

teamId1	מזהה הקבוצה הראשונה.
teamId2	מזהה הקבוצה השנייה.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId1 <=0, teamid2 <= 0, teamId1==teamId2.
FAILURE	אם אין קבוצות במזהים teamId1 או teamId2.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log^* m)$ בממוצע על הקלט משוערך יחד עם unite_by_record ו-update_match.

StatusType unite_by_record(int record)

על מנת להפוך את הליגה להוגנת יותר, מנהלי הליגה שואפים לאחד קבוצות חלשות עם חזקות. לאחר הרצת פקודה זו, אם קיימות <u>בדיוק</u> 2 קבוצות כך שאחת עם מאזן record והשנייה עם מאזן record-, נאחד ביניהן.
בהרחבה – נניח בה"כ שקיימות בדיוק 2 קבוצות במזהים teamId1,teamId2 כך שהמאזן של teamId1 הוא record ושל teamId2 הוא record- (כלומר "מינוס" record). במקרה זה נאחד בין שתי הקבוצות הנ"ל, והמזהה החדש שלהן יהיה teamId1 (כלומר הקבוצה עם המאזן החיובי). באופן דומה לסעיף קודם, לאחר הפעולה לא ניתן להוסיף קבוצה חדשה במזהה teamId2 למבני הנתונים.

<u>פרמטרים</u>:

record	המאזן שלפיו אנו מעוניינים לאחד.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם record <=0.
FAILURE	אם לא קיימים בדיוק 2 קבוצות במזהים teamId1 ו-teamdId2 כך שלאחד מאזן record ולשני מאזן record-.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log^* m)$ בממוצע על הקלט משוערך יחד עם update_match ו-merge_teams.

---

output_t < int > get_jockey_record(int jockeyId)

מחזיר את המאזן של הרוכב במזהה jockeyId.

<u>פרמטרים</u>:

jockeyId	מזהה הרוכב שאת מאזנו יש להחזיר.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם jockeyId <=0.
FAILURE	אם אין סוס במזהה jockeyId.
SUCCESS	במקרה של הצלחה, במקרה זה תוחזר מאזן הרוכב.

<u>סיבוכיות זמן</u>: $O(1)$ בממוצע על הקלט.

output_t < int > get_team_record(int teamId)

מחזיר את המאזן של הקבוצה במזהה teamId. כלומר, מחזיר את סכום הניצחונות של הרוכבים בקבוצה פחות סכום ההפסדים של הרוכבים בקבוצה.

פרמטרים:

teamId	מזהה הקבוצה שאת מאזנה יש להחזיר.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId <=0.
FAILURE	אם אין קבוצה במזהה teamId.
SUCCESS	במקרה של הצלחה, במקרה זה תוחזר גם מאזן הקבוצה.

סיבוכיות זמן: $O(1)$ בממוצע על הקלט.

סיבוכיות מקום:

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא $O(n+m)$ במקרה הגרוע. שימו לב ש-m הוא מספר כלל הקבוצות שנוספו למערכת (כולל קבוצות ש"נמחקו" כתוצאה מאיחוד מוצלח).

סיבוכיות המקום הנדרשת עבור כל פעולה (כלומר, זיכרון "העזר" שכל פעולה משתמשת בו) אינה מצוינת לכל פעולה לחוד, אך אסור לעבור את סיבוכיות המקום הדרושה שמוגדרת לכל המבנה.

ערכי החזרה של הפונקציות:

כל אחת מהפונקציות מחזירה ערך מטיפוס StatusType שייקבע לפי הכלל הבא:

- תחילה, יוחזר INVALID_INPUT אם הקלט אינו תקין.
- אם לא הוחזר INVALID_INPUT:
  - בכל שלב בפונקציה, אם קרתה שגיאת הקצאה/שחרור יש להחזיר ALLOCATION_ERROR. מצב זה אינו צפוי אלא באחד משני מקרים (לרוב): באמת השתמשתם בקלט גדול מאוד ולכן המבנה ניצל את כל הזיכרון במערכת, או שיש זליגת זיכרון בקוד.
  - אם קרתה שגיאה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד FAILURE מבלי לשנות את מבנה הנתונים.
- אחרת, יוחזר SUCCESS.

חלק מהפונקציות צריכות להחזיר בנוסף עוד פרמטר (int או bool), לכן הן מחזירות אובייקט מטיפוס output_t<T>.

אובייקט זה מכיל שני שדות: הסטטוס (__status) ושדה נוסף (__ans) מסוג T.

במקרה של הצלחה (SUCCESS), השדה הנוסף יכיל את ערך החזרה, והסטטוס יכיל את SUCCESS. בכל מקרה אחר, הסטטוס יכיל את סוג השגיאה והשדה הנוסף לא מעניין.

שני הטיפוסים (output_t<T>,StatusType) ממומשים כבר בקובץ "wet2util.h" שניתן לכם כחלק מהתרגיל.

קיים קונסטרקטור של output_t<T> מ-T ומ-StatusType כך שניתן פשוט לכתוב בפונקציות הרלוונטיות:

```
return 7;
return StatusType::FAILURE;
```

הנחיות ודגשים כלליים:

חלק יבש:

- החלק היבש הווה חלק מהציון על התרגיל כפי שמצוין בנהלי הקורס.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- החלק היבש חייב להיות מוקלד.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כלומר, הגשה בה יתקבל אך ורק חלק יבש תגרור ציון 0 על התרגיל כולו.
- יש להכין מסמך הכולל תיאור של מבני הנתונים והאלגוריתמים בהם השתמשתם בצירוף הוכחת סיבוכיות הזמן והמקום שלהם. חלק זה עומד בפני עצמו וצריך להיות מובן לקורא גם לפני העיון בקוד. אין צורך לתאר את הקוד ברמת המשתנים, הפונקציות והמחלקות, אלא ברמה העקרונית. חלק יבש זה לא תיעוד קוד.
- ראשית הציגו את מבני הנתונים בהם השתמשתם. רצוי ומומלץ להיעזר בציור.
- לאחר מכן הסבירו כיצד מימשתם כל אחת מהפעולות הנדרשות. הוכיחו את דרישות סיבוכיות הזמן של כל פעולה תוך כדי התייחסות לשינויים שהפעולות גורמות במבני הנתונים.
- הוכיחו שמבנה הנתונים וכל הפעולות עומדים בדרישת סיבוכיות המקום.
- החסמים הנתונים בתרגיל הם לא בהכרח הדוקים ולכן יכול להיות שקיים פתרון בסיבוכיות טובה יותר. מספיק להוכיח את החסמים הדרושים בתרגיל.
- רמת פירוט: יש להסביר את כל הפרטים שאינם טריוויאליים ושחשובים לצורך מימוש הפעולות ועמידה בדרישות הסיבוכיות. אין לדון בפרטים טריוויאליים (הפעילו את שיקול דעתכם בקשר לזה, ושאלו את האחראי על התרגיל אם אינכם בטוחים). אין לצטט קטעים מהקוד כתחליף להסבר. אין צורך לפרט אלגוריתמים שנלמדו בכתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתכוונים. אין (וגם אין צורך) להשתמש בתוצאות של עצי דרגות והלאה.

- **על חלק זה לא לחרוג מ-8 עמודים.**
- והכי חשוב **keep it simple!**

חלק רטוב:

- מומלץ לממש תחילה את מבני הנתונים בצורה הכללית ביותר ורק אז לממש את הפונקציות הנדרשות בתרגיל.
- אנו ממליצים בחום על מימוש **Object Oriented**, ב**++C**, מימוש כזה יאפשר לכם להגיע לפתרון פשוט וקצר יותר לפונקציות אותן עליכם לממש ויאפשר לכם להכליל בקלות את מבני הנתונים שלכם (זכרו שיש תרגיל רטוב נוסף בהמשך הסמסטר).
- פקודת הקימפול שמורצת בgradescope הינה: g++ -std=c++11 -DNDEBUG -Wall -o main.out *.cpp
- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ plains25a1.h.
- אין לשנות את הקבצים main25a1.cpp ו-wet1util.h אשר סופקו כחלק מהתרגיל, ואין להגיש אותם. ישנה בדיקה אוטומטית שאין בקוד שימוש בSTL, ובדיקה זו נופלת אם מגישים גם את main25a1.cpp.
- את שאר הקבצים ניתן לשנות, ותוכלו להוסיף קבצים נוספים כרצונכם, ולהגיש אותם.
- העיקר הוא שהקוד שאתם מגישים יתקמפל עם הפקודה לעיל, כאשר מוסיפים לו את שני הקבצים main25a1.cpp ו-wet1util.h.
- עליכם לממש בעצמכם את כל מבני הנתונים (למשל **אין להשתמש במבנים של STL** ואין להוריד מבני נתונים מהאינטרנט). **כחלק מתהליך הבדיקה אנו נבצע בדיקה ידנית של הקוד ונוודא שאכן מימשתם את מבני הנתונים שבהם השתמשתם.**
  - בפרט, אסור להשתמש ב-std::iterator, std::vector, std::pair, או כל אלגוריתם של STL, רשימה מלאה של הספריות להן אסור לעשות include נמצאת בקובץ dont_include.txt.
  - **ניתן** להשתמש במצביעים חכמים (Smart pointers כמו shared_ptr), בספריית math או בספריית exception.
  - חשוב לוודא שאתם מקצים/משחררים זיכרון בצורה נכונה (מומלץ לוודא עם valgrind). לא חייבים לעבוד עם מצביעים חכמים, אך אם אתם מחליטים כן לעשות זאת, לוודא שאתם משתמשים בהם נכון. (תזכרו שהם לא פתרון קסם, למשל, כאשר יוצרים מעגל בהצבעות)
  - שגיאות של ALLOCATION_ERROR בד"כ מעידות על זליגה בזיכרון.
  - על הקוד להתקמפל ולעבור את כל הבדיקות שמפורסמות לכם ב-gradescope. הטסטים שמורצים באתר מייצגים את הבדיקת אותן נריץ בנתינת הציון, כאשר פרסמנו 5 מתוך 50.
  - אותם טסטים שבgradescope גם מפורסמים כקבצי קלט ופלט, יחד עם סקריפט בשם run_tests.py שנכתב בשביל python 3.6 ומעלה, המאפשר לבדוק את הקוד שלכם. מומלץ לבדוק את התרגיל לוקאלית לפני שמגישים.
  - **שימו לב:** התוכנית שלכם תיבדק על קלטים רבים ושונים מקבצי הדוגמא הנ"ל. יחד עם זאת הטסטים האלו מייצגים מבחינת אורך ואופן היצירה שלהם את השאר.

אופן ההגשה:

הגשת התרגיל הנה דרך אתר ה-gradescope של הקורס.

חלק הרטוב:

יש להגיש קובץ **ZIP** שמכיל רק את קבצי הקוד שלכם (לרוב קבצי cpp. ,h. בלבד)

חלק היבש:

יש להגיש קובץ PDF אשר מכיל את הפתרון היבש. החלק היבש חייב להיות מוקלד.

- **שימו לב כי אתם מגישים את כל שני החלקים הנ"ל, במטלות השונות.**
- לאחר שהגשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב. ההגשה האחרונה היא הנחשבת.
- הערכת הציון שמופיעה ב-gradescope אינה ציונכם הסופי על המטלה. הציון הסופי יתפרסם רק לאחר ההגשות המאוחרות של משרתי המילואים.
- במידה ואתם חושבים שישנה תקלה מהותית במערכת הבדיקה ב-gradescope נא להעלות זאת בפורום הפיאצה ונתפל בה בהקדם.

דחיות ואיחורים בהגשה:

- דחיות בתרגיל הבית תינתנה אך ורק לפי תקנון הקורס.
- 5 נקודות יורדו על כל יום איחור בהגשה ללא אישור מראש. באפשרותכם להגיש תרגיל באיחור של עד 5 ימים ללא אישור. תרגיל שיוגש באיחור של יותר מ-5 ימים ללא אישור מראש יקבל 0.
- במקרה של איחור בהגשת התרגיל יש עדיין להגיש את התרגיל אלקטרונית דרך אתר הקורס.

- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת jonathan.gal@campus.technion.ac.il. לאחר קבלת אישור במייל על הבקשה, מספר הימים שאושרו לכם נשמר אצלנו. לכן, אין צורך לצרף להגשת התרגיל אישורים נוספים או את שער ההגשה באיחור.

בהצלחה!