מתרגל ממונה על התרגיל: יונתן גל, Jonathan.gal@campus.technion.ac.il

תאריך ושעת הגשה: 27/02/2024 בשעה 23:55

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 "wet_1":
  - האתר: piazza.com/technion.ac.il/winter2024/234218
  - 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.
- 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: barakgahtan@cs.technion.ac.il.

הקדמה:

האולימפיאדה בפריז מתקיימת בקרוב והמארגנים נזכרו רק עכשיו שאין להם מערכת נוחה לניהול התחרות!

לכן הם קידמו את קלוד קרמל קרטיר ממתקן המזגנים של האולימפיאדה למתכנת הראשי. לצערינו קלוד קרמל קרטיר לא למד מבני נתונים ולכן הוא ביקש ממכם עזרה.

כל מדינה, נבחרת, ומתחרה במערכת מיוצגים ע"י מזהה מספרי ייחודי. המערכת מאפשרת הכנסה והוצאה של מדינות, נבחרות, ומתחרים והוספה של מתחרים לנבחרות. המערכת מתחזקת סטטיסטיקות הקשורות למתחרים ולמדינות, ומאפשרת לסמלץ משחקים בין הקבוצות שתוצאותיהם נקבעת לפי הסטטיסטיקה השמורה במערכת.

דרוש מבנה נתונים למימוש הפעולות הבאות:

olympics_t()

מאתחלת מבנה נתונים ריק. תחילה אין במערכת מדינות, נבחרות או מתחרים.

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

---

virtual ~olympics_t()

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

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(n+k+m)$ במקרה הגרוע.

StatusType add_country(int countryId, int medals)

המדינה בעלת מזהה countryId ייחודי משתתפת באלימפיאדה, ולכן צריך להכניס אותה למבנה.

כמות המדליות שהמדינה זכתה בהן בעבר הוא medals.

בעת ההכנסה אין נבחרות או מתחרים השייכים למדינה.

פרמטרים:

countryId	מזהה המדינה החדשה.
medals	כמות המדליות שהמדינה זכתה בהן.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם countryId<=0 או medals<0.
FAILURE	אם countryId הוא מזהה של מדינה קיימת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר המדינות במערכת.

StatusType remove_country(int countryId)

המדינה בעלת מזהה countryId נפסלת מהתחרות, ולכן צריך להוציאה מהמערכת.
אם קיימים מתחרים או נבחרות השייכות למדינה - לא ניתן למחוק אותה, והיא נשארת במערכת; והפעולה נכשלת.

פרמטרים:

countryId	מזהה המדינה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם countryId<=0.
FAILURE	אם אין מדינה בעלת מזהה countryId או שיש נבחרות/מתחרים למדינה.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר המדינות במערכת.

StatusType add_team(int teamId, int countryId, Sport sport)

המדינה countryId פותחת קבוצה בעלת מזהה ייחודי teamId לספורט sport. בהתחלה אין לנבחרת מתחרים.
לכל מדינה יכולות להתקיים כמה נבחרות מכל סוג.

*הערה: הטיפוס Sport מוגדר עבורכם בקובץ h.util1wet והוא מקבל את הערכים
BOULDERING, ACROBATICS, FOOTBALL, SWIMMING, SPINNING

פרמטרים:

teamId	מזהה הנבחרת שצריך להוסיף.
countryId	מזהה המדינה של הנבחרת.
sport	סוג הספורט של הנבחרת.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId<=0, countryId<=0.
FAILURE	אם קיימת כבר נבחרת עם מזהה teamId, שהמדינה עם המזהה countryId לא קיימת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log m + \log k)$ במקרה הגרוע, כאשר m הוא מספר הנבחרות במערכת, ו-k הוא מספר המדינות במערכת.

StatusType remove_team(int teamId)

הנבחרת בעלת מזהה teamId מורחקת מהאולימפיאדה, ולכן צריך להוציאה מהמערכת.
אם קיימים מתחרים השייכים לנבחרת - לא ניתן למחוק אותה, והיא נשארת במערכת; והפעולה נכשלת.

פרמטרים:

teamId	מזהה הנבחרת.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId<=0.
FAILURE	אם אין נבחרת עם מזהה teamId או שיש מתחרים בנבחרת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log m)$ במקרה הגרוע, כאשר m הוא מספר הנבחרות במערכת.

StatusType add_contestant(int contestantId , int countryId, Sport sport, int strength)

מתחרה בעל מזהה ייחודי contestantId משתתף באולימפיאדה עבור מדינה countryId.
המתחרה משתתף בספורט sport, והכח שלו הוא strength.

פרמטרים:

contestantId	מזהה המתחרה שצריך להוסיף.
countryId	מזהה המדינה של המתחרה.
sport	סוג הספורט שהמתחרה משחק.
strength	הכח ההתחלתי של המתחרה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם contestantId<=0, countryId<=0, strength<0
FAILURE	אם קיים כבר מתחרה עם מזהה contestantId או שהמדינה עם המזהה countryId לא קיימת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log n + \log k)$ במקרה הגרוע, כאשר n הוא מספר המתחרים במערכת, ו-k הוא מספר המדינות במערכת.

StatusType remove_contestant(int contestantId)

המתחרה בעל מזהה contestantId התגייס למילואים, ולכן לא יוכל להמשיך להשתתף בתחרות, ולכן צריך להוציאו ממבנה הנתונים. לא ניתן להוציא מתחרה אם הוא כרגע פעיל בנבחרת.

פרמטרים:

contestantId	מזהה המתחרה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם contestantId<=0.
FAILURE	אם אין מתחרה עם מזהה contestantId או שהמתחרה פעיל בנבחרת כלשהיא.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log n)$ במקרה הגרוע, כאשר n הוא מספר המתחרים במערכת.

StatusType add_contestant_to_team(int teamId, int contestantId)

המתחרה contestantId מצטרף לנבחרת teamId. המדינה וסוג הספורט של המתחרה והנבחרת צריכים להיות זהים. מתחרה יכול להיות חלק מכמה נבחרות. כל מתחרה יכול להשתתף בעד ל3 נבחרות (אחת לכל מדליה).

פרמטרים:

teamId מזהה הנבחרת.

contestantId מזהה המתחרה שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT אם teamId<=0, countryId<=0.

FAILURE אם המדינה או הספורט של המתחרה והנבחרת לא מתאימים, שהמתחרה כבר בנבחרת או שהמתחרה כבר ב3 נבחרות שונות.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(\log n + \log m)$ במקרה הגרוע, כאשר m הוא מספר הנבחרות במערכת, ו-n הוא מספר המתחרים במערכת.

StatusType remove_contestant_from_team(int teamId, int contestantId)

השחקן contestantId פורש מהנבחרת בעלת מזהה teamId.

פרמטרים:

teamId מזהה השחקן.

ערך החזרה:

ALLOCATION_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT אם teamId<=0.

FAILURE אם אין נבחרת עם מזהה teamId, אין מתחרה עם מזהה contestantId, או שcontestantId לא נמצא בteamId.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(\log n + \log m)$ במקרה הגרוע, כאשר m הוא מספר הנבחרות במערכת, ו-n הוא מספר המתחרים במערכת.

StatusType update_contestant_strength(int contestantId , int change)

בגלל נסיבות החיים, המתחרה בעל מזהה contestantId מתחזק או נחלש. תשנו את מדד הstrength של המתחרה במערכת בchange. הכח של מתחרה לא יכול לרדת מתחת ל0.

פרמטרים:

contestantId מזהה המתחרה.

strength הכח החדש של המתחרה.

ערך החזרה:

ALLOCATION_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT אם contestantId<=0.

FAILURE אם אין מתחרה עם מזהה contestantId או שהכח שלו אחרי השינוי קטן מ0

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(\log n + \log m)$ במקרה הגרוע, כאשר n הוא מספר המתחרים במערכת וm הוא מספר הנבחרות במערכת.

---

output_t < int > get_strength(int contestantId)

יש להחזיר את הכוח של המתחרה contestantId.

<u>פרמטרים:</u>

contestantId	מזהה המתחרה.

<u>ערך החזרה:</u> מספר המשחקים הכולל בהם השתתף השחקן, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם contestantId <=0.
FAILURE	אם אין מתחרה עם מזהה contestantId.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן:</u> $O(\log n)$ במקרה הגרוע, כאשר n הוא מספר המתחרים במערכת.

---

output_t < int > get_medals(int countryId)

יש להחזיר את כמות המדליות של המדינה countryId.

<u>פרמטרים:</u>

countryId	מזהה המדינה.

<u>ערך החזרה:</u> מספר המדליות הכולל של המדינה, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם countryId <=0.
FAILURE	אם אין מדינה עם מזהה countryId.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן:</u> $O(\log k)$ במקרה הגרוע, כאשר k הוא מספר המדינות במערכת.

---

output_t < int > get_team_strength(int teamId)

חישוב הכח של שחקני נבחרת מתבצע בצרה הבאה:

אם מספר המשתתפים בנבחרת teamId לא מתחלק ב3 אז הכוח של הנבחרת הוא 0.

אחרת, נכתוב את כמות המתחרים בקבוצה כS * 3, ונסדר את המתחרים לפי הcontestantId שלהם. ניקח את הכח של המתחרה הכי חזק מבין S המתחרים עם הcontestantId הכי קטן, נוסיף לו את הכוח של המתחרה הכי חזק מבין הS הבאים, ואז נוסיף לסכום את הכוח של המתחרה הכי חזק מבין S המתחרים עם הcontestantId הכי גדול.

לדוגמא אם יש 9 שחקנים עם contestantId

[1, 4, 6, 10, 16, 20, 100, 101, 102]

ועם כוחות

[2, 5, 3, 10, 11,15, 7, 6, 4]

אז הכח של הנבחרת הוא 27 (7 + 15 + 5)

עוד דוגמה אם יש 12 שחקנים עם contestantId

[1,2,3,4,10,20,30,40,100,200,300,400]

ועם כוחות

[800,700,600,500,80,70,60,50,8,7,6,5]

אז הכח של הנבחרת הוא 888 (400 + 40 + 4)

<u>פרמטרים:</u>

teamId	מזהה הקבוצה.

<u>ערך החזרה:</u> הכח הכולל של הנבחרת, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId<=0.
FAILURE	אם אין נבחרת עם מזהה teamId.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן:</u> $O(\log m)$ במקרה הגרוע, כאשר m הוא מספר הנבחרות במערכת.

StatusType unite_teams(int teamId1, int teamId2)

הנבחרות teamId1 ו-teamId2 מחליטות לאחד כוחות ולהשתתף תחת קבוצה אחת מאוחדת בעלת המזהה teamId1.

הקבוצה החדשה תכיל את כל השחקנים של שתי הקבוצות המקוריות, והניקוד שלה יהיה סכום הניקוד של שתיהן.

ניתן לאחד רק בין קבוצות באותו ספורט שמשחקות עבור אותה מדינה. אם אותו מתחרה נמצא בשתי הנבחרות הוא לא משתכפל אלא נמצא רק פעם אחת בקבוצה החדשה.

<u>פרמטרים</u>:

teamId1	מזהה קבוצה ראשונה.
teamId2	מזהה קבוצה שניה.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId1 <=0, teamId2<=0, teamId2==teamId1
FAILURE	אם אין קבוצות עם המזהה teamId2/teamId1 , או המדינה/הספורט של הנבחרות לא זהה.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log m + n_{Team1ID} + n_{Team2ID})$ במקרה הגרוע, כאשר m הוא מספר הנבחרות.

$n_{Team1ID}$ ו- $n_{Team2ID}$ הם מספרי המתחרים בכל אחת מהקבוצות המקוריות.

StatusType play_match(int teamId1, int teamId2)

שתי הנבחרות בעלות המזהים teamId1 ו-teamId2 משחקות אחת מול השנייה בתחרות. כדי לשחק שתי הנבחרות צריכות להשתתף באותו הספורט.

הניקוד של נבחרת הוא כמות המדליות במדינה שהנבחרת משחקת בשבילה ועוד הכח של הנבחרת שחושב בget_team_strength

אם שני ניקודים שווים, אז המשחק מסתיים בתיקו, אחרת המדינה של הנבחרת המנצחת מקבלת מדליה.

נבחרות באותה מדינה יכולות לשחק אחת נגד השנייה (ואפילו מתחרה יכול להתחרות נגד עצמו!)

<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, teamId2==teamId1
FAILURE	אם אין קבוצה עם מזהה teamId1 או teamId2. או ששתי הנבחרות משחקות בענפי ספורט שונים.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log k + \log m)$ במקרה הגרוע, כאשר m הוא מספר הנבחרות, וk מספר המדינות.

---

output_t < int > austerity_measures(int teamId)

הנבחרת teamId רוצה לקצץ 3 מתחרים שלה, אך לא יודעת באיזה מתחרים לבחור! הפונקצייה מחזירה את כמות הכח המקסימלית שיכולה להיות לנבחרת אחרי העפה של 3 מתחרים.

הערה: הפונקציה רק בודקת את הכח המקסימלי ולא מעיפה שחקנים בעצמה!

לדוגמא אם יש 9 שחקנים עם contestantId

[1, 4, 6, 10, 16, 20, 100, 101, 102]

ועם כוחות

[2, 5, 3, 10, 11,15, 7, 6, 4]

אז ערך החזרה של austerity_measures על הנבחרת הוא 32 (10 + 15 + 7 או 6+15+11)

<u>פרמטרים</u>:

teamId מזהה הנבחרת.

<u>ערך החזרה</u>: המזהה של השחקן שהבקיע הכי הרבה שערים, ובנוסף סטטוס:

ALLOCATION_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT אם teamId<=0.

FAILURE אם אין נבחרת בשם teamId או שיש פחות מ3 שחקנים בteamId.

SUCCESS במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log m)$ במקרה הגרוע, כאשר m הוא מספר הנבחרות במערכת.

<u>סיבוכיות מקום</u>:

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא $O(n+k+m)$ במקרה הגרוע, כאשר n הוא מספר השחקנים , k הוא מספר הקבוצות וm הוא מספר המדינות. כלומר בכל רגע בזמן הריצה, צריכת המקום של מבנה הנתונים תהיה לינארית בסכום מספרי השחקנים, הקבוצות והמדינות במערכת.

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.

<u>ערכי החזרה של הפונקציות:</u>

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 <u>מבלי</u> לשנות את מבנה הנתונים.
- אחרת, יוחזר SUCCESS.

חלק מהפונקציות צריכות להחזיר בנוסף עוד פרמטר (לרוב int), 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 "wet1util.h" שניתן לכם כחלק מהתרגיל.

הנחיות:

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.
- 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.
- **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).
- על הקוד להתקמפל על csl3 באופן הבא:

g++ -std=c++11 -DNDEBUG -Wall *.cpp

עליכם מוטלת האחריות לוודא קומפילציה של התכנית ב+ + g. אם בחרתם לעבוד בקומפיילר אחר, מומלץ לקמפל ב+ + g מידי פעם במהלך העבודה.

הערות נוספות:

- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ olympics24a1.h.
- קראו היטב את הקובץ הנ"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים main24a1.cpp ו-wet1util.h אשר סופקו כחלק מהתרגיל, **ואין להגיש אותם**.
  - את שאר הקבצים ניתן לשנות.
  - תוכלו להוסיף קבצים נוספים כרצונכם, ולהגיש אותם.
  - העיקר הוא שהקוד שאתם מגישים יתקמפל עם הפקודה לעיל, כאשר מוסיפים לו את שני הקבצים main24a1.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.

- **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.
- מצורפים לתרגיל קבצי קלט ופלט לדוגמא, ניתן להריץ את התוכנה על הקלט ולהשוות עם הפלט המצורף.
- **שימו לב:** התוכנית שלכם תיבדק על קלטים שונים מקבצי הדוגמא הנ"ל, שיהיו ארוכים ויכללו מקרי קצה שונים. לכן, מומלץ **מאוד** לייצר בעצמכם קבצי קלט, לבדוק את התוכנית עליהם, ולוודא שהיא מטפלת נכון בכל מקרה הקצה.

הגשה:

- **חלק יבש+ חלק רטוב**:

הגשת התרגיל הנה **אך ורק** אלקטרונית דרך אתר הקורס.
יש להגיש קובץ **ZIP** שמכיל את הדברים הבאים:

- בתיקייה הראשית:
  - קבצי ה-Source Files שלכם. **למעט הקבצים main24a1.cpp ו-wet1util.h**, שאסור לשנות.
  - קובץ PDF בשם dry.pdf אשר מכיל את הפתרון היבש. מומלץ להקליד את החלק הזה אך ניתן להגיש קובץ PDF מבוסס על סריקה של פתרון כתוב בכתב יד. שימו לב כי במקרה של כתב לא קריא, כל החלק השני לא תיבדק.
  - קובץ submissions.txt, המכיל בשורה הראשונה את שם, תעודת הזהות וכתובת הדוא"ל של השותף הראשון ובשורה השנייה את שם, תעודת הזהות וכתובת הדוא"ל של השותף השני. לדוגמה:

John Doe 012345678 doe@cs.technion.ac.il
Henry Taub 123456789 taub@cs.technion.ac.il

- **שימו לב כי אתם מגישים את כל שלושת החלקים הנ"ל.**
- אין להשתמש בפורמט כיווץ אחר (לדוגמה RAR), מאחר ומערך הבדיקה האוטומטי אינו יודע לזהות פורמטים אחרים.
- **יש לוודא שכאשר נכנסים לקובץ הזיפ הקבצים מופיעים מיד בתוכו ולא בתוך תיקיה שבתוך קובץ הזיפ.** עבור הגשה שבה הקבצים יהיו בתוך תיקייה, הבדיקה האוטומטית לא תמצא את הקבצים ולא תוכל לקמפל ולהריץ את הקוד שלכם ולכן תיתן אוטומטית 0.
- לאחר שהגשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב. ההגשה האחרונה היא הנחשבת.
- הגשה שלא תעמוד בקריטריונים הנ"ל תפסל ותקנס בנקודות!
  - אחרי שאתם מכינים את ההגשה בקובץ zip מומלץ מאוד לקחת אותה לשרת ולהריץ את הבדיקות שלכם עליה כדי לוודא שאתם מגישים את הקוד שהתכוונתם להגיש בדיוק (ושהוא מתקמפל).

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 barakgahtan@cs.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!