מתרגל ממונה על התרגיל: שחר כהן, shacharcohen@campus.technion.ac.il

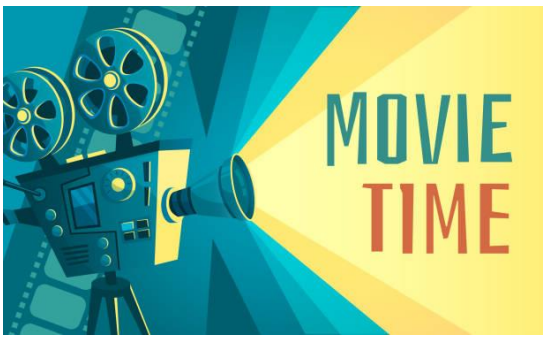
תאריך ושעת הגשה: TBD בשעה 23:55

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/spring2023/234218
  - 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 למתרגלת האחראית בלבד בכתובת: turutovsally@campus.technion.ac.il.

הקדמה:



עקב הגדלת בתחרות בעולם שירותי הסטרימינג, שירות הסטרימינג Webflix רוצה לשפר את השירות שלו. לצורך כך השירות ביקש את עזרתכם בבניית מבנה נתונים שיאפשר לו לנהל בצורה טובה ויעילה יותר את מאגר הסרטים והמשתמשים שלו.

כל משתמש או סרט במערכת מיוצגים ע"י מזהה מספרי ייחודי.

המערכת מאפשרות הכנסה והוצאה של משתמשים וסרטים, וכן מאפשרת מעקב אחרי סטטיסטיקות שונות הקשורות למשתמשים ולסרטים. בנוסף המערכת תאפשר למשתמשים לצפות בסרטים לבד או בקבוצות צפייה ותדע לספק המלצות לצפייה.

דרוש מבנה נתונים למימוש הפעולות הבאות:

streaming_database()

מאתחלת מבנה נתונים ריק. תחילה אין במערכת סרטים או משתמשים.

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

---

virtual ~streaming_database()

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

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן: $O(n+k+m)$ במקרה הגרוע, כאשר n הוא מספר המשתמשים הכולל, k הוא מספר הסרטים ו- m הוא מספר קבוצות הצפייה במערכת.

StatusType add_movie(int movieId, Genre genre, int views, bool vipOnly)

הסרט בעל המזהה movieId התווסף לשירות ולכן מוכנס למבנה הנתונים.

הז'אנר של הסרט מוגדר על ידי genre והוא יכול להיות רק מתוך רשימה מוגדרת מראש ו views הוא מספר הצפיות ההתחלתי של הסרט.

השדה הבוליאני vipOnly מציין האם הסרט זמין רק למשתמשי vip.

***הערה:** הטיפוס Genre מוגדר עבורכם בקובץ wet1util.h והוא מקבל את הערכים: COMEDY, DRAMA, ACTION, FANTASY, NONE, אתם יכולים להניח שערכים אלה לא ישתנו בשום שלב.

פרמטרים:

movieId	מזהה הקבוצה החדשה.
genre	הז'אנר של הסרט.
views	מספר הצפיות ההתחלתי של הסרט.
vipOnly	האם הסרט זמין רק למשתמשי vip

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם movieId<=0, genre=NONE, או views<0.
FAILURE	אם movieId הוא מזהה של סרט קיים.
SUCCESS	במקרה של הצלחה.

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

StatusType remove_movie(int movieId)

הסרט בעל המזהה movieId מורד מהשירות ולכן יוצא מהמערכת.

פרמטרים:

movieId	מזהה הסרט.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם movieId<=0.
FAILURE	אם אין סרט בעל מזהה movieId.
SUCCESS	במקרה של הצלחה.

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

StatusType add_user(int userId, bool isVip)

המשתמש בעל מזהה ייחודי userId נרשם לשירות ולכן צריך להוסיף אותו למערכת.

המשתנה isVip קובע האם המשתמש הוא משתמש vip.

פרמטרים:

userId	מזהה המשתמש שצריך להוסיף.
isVip	האם המשתמש הוא vip.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם userId<=0.
FAILURE	אם קיים כבר משתמש עם מזהה userId.
SUCCESS	במקרה של הצלחה.

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

StatusType remove_user(int userId)

המשתמש בעל מזהה userId עזב את השירות, ולכן צריך להוציאו ממבנה הנתונים, אם הוא בקבוצה צריך להוציא אותו מהקבוצה.

פרמטרים:

userId	מזהה המשתמש.

ערך החזרה:

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

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

StatusType add_group(int groupId)

הוספת קבוצה למערכת לצורך צפייה משותפת של מספר משתמשים, הקבוצה החדשה מתווספת כשהיא ריקה.

פרמטרים:

groupId	מזהה הקבוצה שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם groupId<=0.
FAILURE	אם קיימת כבר קבוצה עם מזהה groupId.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(logm)$ במקרה הגרוע, כאשר m הוא מספר הקבוצות במערכת.

StatusType remove_group(int groupId)

הקבוצה בעלת מזהה groupId פורקה, ולכן צריך להוציא אותה ממבנה הנתונים.

פרמטרים:

groupId	מזהה הקבוצה.

ערך החזרה:

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

סיבוכיות זמן: $O(logm + n_{groupUsers})$ במקרה הגרוע, כאשר m הוא מספר הקבוצות במערכת ו- $n_{groupUsers}$ הוא מספר המשתמשים בקבוצה.

StatusType add_user_to_group(int userId, int groupId)

המשתמש בעל מזהה userId הצטרף לקבוצת צפייה בעלת מזהה groupId, כל משתמש יכול להיות רק בקבוצת צפייה אחת.

***הערה:** אין פונקציה שמסירה משתמש מקבוצה, המשמעות היא שהדרך היחידה להסיר משתמש מקבוצה (בלי למחוק אותו מהמערכת) היא לפרק את הקבוצה שבה הוא נמצא.

פרמטרים:

userId	מזהה המשתמש.
groupId	מזהה הקבוצה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם userId <=0, groupId <=0.
FAILURE	אם אין משתמש עם מזהה userId, אין קבוצה עם מזהה groupId או שהמשתמש כבר נמצא בקבוצת צפייה אחרת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log n + logm)$ במקרה הגרוע, כאשר n הוא מספר המשתמשים ו m הוא מספר הקבוצות במערכת.

StatusType user_watch(int userId, int movieId)

המשתמש בעל המזהה הייחודי userId רוצה לצפות בסרט בעל המזהה הייחודי movieId, אם הסרט מוגדר למשתמשי vip בלבד והמשתמש הוא לא משתמש vip אז הפעולה אסורה.

פרמטרים:

userId	מזהה המשתמש.
movieId	מזהה הסרט.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם userId<=0 או movieId<=0.
FAILURE	אם אין משתמש עם מזהה userId, אין סרט עם מזהה movieId או שהסרט מוגדר למשתמשי vip בלבד והמשתמש הוא לא משתמש vip.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(logn + \log k)$ במקרה הגרוע, כאשר n הוא מספר המשתמשים ו- k הוא מספר הסרטים.

StatusType group_watch(int groupId, int movieId)

הקבוצה בעלת המזהה הייחודי groupId רוצה לצפות בסרט בעל המזהה הייחודי movieId, מספר הצפיות של הסרט יגדל לפי מספר המשתמשים בקבוצה. קבוצה ריקה לא יכולה לצפות בסרטים ואם הסרט מוגדר למשתמשי vip בלבד רק קבוצה שהיא קבוצת vip יכולה לצפות בו. קבוצה תוגדר קבוצת vip אם יש בה משתמשי vip (מותר שיהיו גם משתמשים שהם לא vip)

<u>פרמטרים</u>:

groupId	מזהה הקבוצה.
movieId	מזהה הסרט.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם groupId<=0 או movieId<=0.
FAILURE	אם אין קבוצה עם מזהה groupId, אין סרט עם מזהה movieId, הקבוצה קיימת אבל ריקה או שהסרט מוגדר למשתמשי vip בלבד והקבוצה היא לא קבוצת vip.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(logm + \log k)$ במקרה הגרוע, כאשר m הוא מספר הקבוצות ו- k הוא מספר הסרטים.

---

output_t < int > get_all_movies_count(Genre genre)

אם genre=NONE, הפעולה תחזיר את מספר הסרטים במערכת.
אחרת, הפעולה תחזיר את מספר הסרטים בז'אנר המבוקש.

<u>פרמטרים</u>:

genre	הז'אנר המבוקש.

<u>ערך החזרה</u>: מספר הסרטים בז'אנר או בכל המערכת, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(1)$ במקרה הגרוע.

StatusType get_all_movies(Genre genre, int * const output)

כדי לאפשר למשתמשים לבחור סרט נרצה פונקציה שמחזירה את כל הסרטים במערכת או בז'אנר מסויים.
אם genre=NONE, הפעולה ממלאת במערך output את ה movieId של כל הסרטים במערכת.
אחרת, הפעולה ממלאת במערך output את ה movieId של כל הסרטים מהז'אנר המבוקש.
בשני המקרים, המזהים של הסרטים יהיו ממוינים בסדר יורד לפי הדירוג הממוצע של הסרט מכל המשתמשים במערכת כאשר אם הסרט לא דורג על ידי אף משתמש נגדיר שהדירוג שלו הוא 0, במקרה של שוויון בסדר יורד לפי מספר הצפיות, ובמקרה של שוויון נוסף בסדר עולה לפי movieId.
output מצביע למקום הראשון במערך הפלט. המערך מוקצה (בגודל המתאים) ומשוחרר בקוד שקורא לפונקציה זו.
הגודל המתאים זה הערך המוחזר מ-get_all_movies _count עבור אותו הקלט, במקרה של הצלחה.

<u>פרמטרים</u>:

genre	הז'אנר המבוקש.
output	מערך המזהים המוחזר במקרה הצלחה.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם output==NULL.
FAILURE	אם genre!=NONE ואין סרטים בז'אנר זה, או אם genre=NONE ואין סרטים במערכת.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: אם genre=NONE, אז $O(k)$ במקרה הגרוע, כאשר k הוא מספר הסרטים במערכת.
אחרת, $O(k_{genre})$ במקרה הגרוע, כאשר k_{genre} הוא מספר הסרטים בז'אנר המבוקש.

---

output_t < int > get_num_views(int userId, Genre genre)

מפתחי השירות רוצים לעקוב אחרי הרגלי הצפייה של משתמשים ולצורך כך הם רוצים לדעת בכמה סרטים מז'אנר מסויים או מהמערכת צפו משתמשים שונים, כאשר כל צפייה נוספת של משתמש מסויים באותו סרט נספרת בנפרד.
לדוגמא אם אבי צפה 15 פעמים בסרט "Kill Bill 1" ולא צפה באף סרט אחר ונועה צפתה פעם אחת בסרט "Kill Bill 1" ופעם אחת בסרט "Pulp Fiction" אז נגדיר שאבי צפה ב 15 סרטים ונועה צפתה בשני סרטים.
אם genre=NONE הפעולה תחזיר את מספר הצפיות של המשתמש עם המזהה userID בכל הסרטים במערכת.
אחרת הפעולה תחזיר את מספר הצפיות של המשתמש עם המזהה userID רק בז'אנר הספציפי.
במקרה של שוויון נחזיר את המשתמש עם המזהה הקטן ביותר.

<u>פרמטרים</u>:

userID	מזהה המשתמש.
Genre	הז'אנר המבוקש

<u>ערך החזרה</u>: מזהה המשתמש שצפה בהכי הרבה סרטים, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם userID<=0.
FAILURE	אם userID >0 ואין משתמש עם מזהה זה.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(logn)$ במקרה הגרוע כאשר n הוא מספר המשתמשים במערכת.

StatusType rate_movie(int userId, int movieId, int rating)

המשתמש בעל המזהה userId רוצה לתת דירוג של מס' נקודות לפי rating לסרט בעל המזהה movieId

<u>פרמטרים</u>:

userId	מזהה המשתמש.
movieId	מזהה הסרט.
Rating	הדירוג הניתן.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם userId<=0, movieId<=0, rating<0 או rating>100.
FAILURE	אם אין משתמש עם מזהה userId, אין סרט עם מזהה movieId או שהסרט הוא למשתמשי vip בלבד והמשתמש הוא לא משתמש vip.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log n + logk)$ במקרה הגרוע, כאשר n הוא מספר המשתמשים ו k הוא מספר הסרטים במערכת.

---

output_t < int > get_group_recommendation(int groupId)

הקבוצה בעלת המזהה groupId מתקשה להחליט באיזה סרט לצפות ורוצה לקבל המלצה מהמערכת. ההמלצה נקבעת על פי הסרט עם הדירוג הממוצע הכי גבוה ממשתמשי המערכת בז'אנר האהוב על הקבוצה, שנקבע להיות הז'אנר שסך הצפיות בו של כל חברי הקבוצה הוא הכי גדול.
לדוגמא אם רון שלומי ודנה בקבוצה ביחד, רון צפה 20 פעמים בסרטי קומדיה (יכול להיות שכל ה 20 צפיות הן באותו סרט) ו 3 פעמים בסרטי פנטזיה, דנה צפתה פעם אחת בסרט דרמה, ושלומי צפה פעם אחת בסרט דרמה, ובנוסף הם ביצעו צפייה קבוצתית ב 5 סרטי פנטזיה, אז סך הצפיות לסרטי קומדיה הוא 20, סך הצפיות לסרטי פנטזיה הוא 18 וסך הצפיות לסרטי דרמה הוא 2 אז הז'אנר האהוב על הקבוצה הוא קומדיה.
במקרה של שוויון בין ז'אנרים שאהובים על הקבוצה, הז'אנר המנצח יהיה זה שמוגדר קודם בטיפוס Genre, ובמקרה של שוויון בדירוג הסרט הנבחר יהיה זה עם מספר הצפיות הגדול ביותר ובמקרה של שוויון נוסף יבחר הסרט עם המזהה movieId הקטן יותר.

<u>פרמטרים</u>:

groupId	מזהה הקבוצה.

<u>ערך החזרה</u>: מזהה הסרט שניתן כהמלצה לקבוצה, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT	אם groupId<=0.
FAILURE	אם אין קבוצה עם מזהה groupId או שהקבוצה קיימת אבל ריקה, או אם אין סרטים בז'אנר הנבחר.
SUCCESS	במקרה של הצלחה.

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

<u>סיבוכיות מקום</u>:
סיבוכיות המקום הדרושה עבור מבנה הנתונים היא $O(n+k+m)$ במקרה הגרוע, כאשר n הוא מספר המשתמשים, k הוא מספר הסרטים ו- m הוא מספר הקבוצות. כלומר בכל רגע בזמן הריצה, צריכת המקום של מבנה הנתונים תהיה לינארית בסכום מספרי המשתמשים הסרטים והקבוצות 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 וטיפוסים נוספים:</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.
בנוסף בחלק מהפונקציות אתם מקבלים או משתמשים במשתנה מטיפוס Genre, משתנה זה יכול לקבל את אחד מהערכים הבאים: COMEDY, DRAMA, ACTION, FANTASY, NONE.
שלושת הטיפוסים (StatusType,output_t<T>, Genre) 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 worldcup23a1.h.
- קראו היטב את הקובץ הנ"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים main23a1.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 main23a1.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 main23a1.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 למתרגלת האחראית בלבד בכתובת turutovsally@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!