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

תאריך ושעת הגשה: 27/03/2024 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_2״:
  - 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:

לאחר שהאולימפיאדה בפריז כבר התחילה והמתכנת הראשי קלוד קרמל קרטיר השתמש במערכת שפיתחתם עבורו בתרגיל מס' 1, הוא נזכר שיש עוד תחום ספורט נוסף עם חוקים שונים מאלה שהמערכת הקודמת תומכת בהם לכן הוא ביקש מכם לבנות מערכת נוספת עבור התחום הנוסף. המערכת תתמוך בהוספה והסרה של קבוצות ושחקנים חדשים, ובנוסף המערכת תתמוך בתחזוקה של סטטיסטיקות ובסימלוץ של משחקים בודדים וגם טורנירים בין הקבוצות.

ממשו מבנה נתונים התומך בפעולות הבאות:
(לאורך כל השאלה n הינו מספר הקבוצות ו k הינו מספר השחקנים במערכת.

olympics_t()

מאתחלת מבנה נתונים ריק. תחילה אין במערכת קבוצות או שחקנים.
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: $O(n+k)$ במקרה הגרוע, כאשר n הוא מספר הקבוצות במערכת ו-k הוא מספר השחקנים.

StatusType add_team(int teamId)

הקבוצה בעלת מזהה ייחודי teamId משתתפת בתחרות, ולכן צריך להוסיפה למבנה הנתונים.
בעת ההכנסה אין שחקנים בקבוצה.
פרמטרים:

teamId	מזהה הקבוצה החדשה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId ≤ 0.
FAILURE	אם teamId הוא מזהה של קבוצה קיימת.
SUCCESS	במקרה של הצלחה.

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

StatusType remove_team(int teamId)

הקבוצה בעלת המזהה teamId מודחת מהטורניר, ולכן צריך להוציאה מהמערכת, יחד עם כל שחקניה.
לאחר המחיקה, יתכן שתתווסף למבנה קבוצה **אחרת** בעלת אותו מזהה.

פרמטרים:

teamId	מזהה הקבוצה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $teamId \leq 0$.
FAILURE	אם אין קבוצה בעלת מזהה teamId.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log(n) + k_{players\ in\ team})$ משוערך, כאשר n הוא מספר הקבוצות במערכת ו $k_{players\ in\ team}$ הוא מספר השחקנים בקבוצה.

StatusType add_player(int teamId, int playerStrength)

שחקן חדש הצטרף לקבוצה בעלת המזהה teamId בתחרות. הכוח שלו נתון על ידי playerStrength שחייב להיות גדול ממש מ-0.

הבהרה: לשחקנים אין מזהה ייחודי, אתם יכולים להניח שאם הקריאה לפונקציה היא חוקית השחקן לא קיים כבר בקבוצה או בכל קבוצה אחרת.

פרמטרים:

teamId	מזהה הקבוצה של השחקן.
playerStrength	הכוח של השחקן.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	$teamId \leq 0$ או $playerStrength \leq 0$.
FAILURE	אם הקבוצה עם המזהה teamId לא קיימת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log(n) + \log(k_{players\ in\ team}))$ במקרה הגרוע כאשר n הוא מספר הקבוצות במערכת ו $k_{players\ in\ team}$ הוא מספר השחקנים בקבוצה.

StatusType remove_newest_player(int teamId)

הקבוצה בעלת המזהה teamId מדיחה את השחקן החדש ביותר.

פרמטרים:

teamId	מזהה הקבוצה של השחקן.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $teamId \leq 0$.
FAILURE	אם הקבוצה עם המזהה teamId לא קיימת או שהיא ריקה.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log(n) + \log(k_{players\ in\ team}))$ במקרה הגרוע כאשר n הוא מספר הקבוצות במערכת ו $k_{players\ in\ team}$ הוא מספר השחקנים בקבוצה.

---

output_t<int> play_match(int teamId1, int teamId2)

שתי הקבוצות בעלות המזהים teamId1 ו-teamId2 משחקות אחת מול השנייה.
הכוח של כל קבוצה נקבע לפי הכוח של השחקן החציוני מבחינת הכוח בקבוצה (כלומר השחקן הכי חלש שחזק מ $\left\lceil \frac{k_{players\ in\ team}}{2} \right\rceil$ השחקנים החלשים ביותר בקבוצה כאשר $k_{players\ in\ team}$ הוא מספר השחקנים בקבוצה) כפול גודל הקבוצה, והקבוצה החזקה יותר מבין השתיים. במקרה של תיקו בכוחות הקבוצה עם המזהה הקטן יותר תנצח.
מס' הניצחונות של הקבוצה המנצחת גדל ב-1.

לדוגמא אם קבוצה 3 היא עם שחקנים בעלי הכוחות: (1,3,3,5,7,8) וקבוצה 8 היא עם שחקנים בעלי הכוחות (2,4,3,6,6) אז הכוח של השחקן החציוני בקבוצה 3 הוא 5 לכן כוח הקבוצה הוא $5 \cdot 6 = 30$ והכוח של השחקן החציוני בקבוצה 8 הוא 4 לכן כוח הקבוצה הוא $4 \cdot 5 = 20$ לכן קבוצה 3 תנצח.

<u>פרמטרים</u>:

teamId1	מזהה הקבוצה הראשונה.
teamId2	מזהה הקבוצה השנייה.

<u>ערך החזרה</u>: מזהה הקבוצה המנצחת. ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $\text{teamId1} \le 0, \text{teamId2} \le 0$ או $\text{teamId1} = \text{teamId2}$.
FAILURE	אם אין קבוצה עם מזהה teamId1 או teamId2 או שאחת הקבוצות ריקה.
SUCCESS	במקרה של הצלחה.

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

---

output_t<int> num_wins_for_team(int teamId)

יש להחזיר את מספר הניצחונות הכולל של הקבוצה בעלת מזהה teamId מבין המשחקים בהם היא השתתפה.

<u>פרמטרים</u>:

teamId	מזהה הקבוצה.

<u>ערך החזרה</u>: מספר המשחקים הכולל בהם השתתפה הקבוצה, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $\text{teamId} \le 0$.
FAILURE	אם הקבוצה עם המזהה teamId לא קיימת.
SUCCESS	במקרה של הצלחה.

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

---

output_t<int> get_highest_ranked_team()

יש להחזיר את הדירוג של הקבוצה בעלת הדירוג הכי גבוה במערכת. הדירוג נקבע על פי כוח הקבוצה ועוד מספר הנצחונות.
כוח של קבוצה מוגדר באותה צורה כמו בפונקציה play_match.

<u>ערך החזרה</u>: הדירוג של הקבוצה בעלת הדירוג הכי גבוה, ובנוסף סטטוס:

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

<u>סיבוכיות זמן</u>: $O(1)$ במקרה הגרוע, כאשר n מספר הקבוצות במערכת.

StatusType unite_teams(int teamId1, int teamId2)

הקבוצה בעלת teamid1 רוכשת את הקבוצה בעלת teamid2, כתוצאה מכך הקבוצה בעלת teamid2 מפורקת וכל השחקנים שלה עוברים לקבוצה בעלת teamid1 שתישאר במערכת עם אותו המזהה, הכוח של כל השחקנים נשאר אותו הדבר ומספר המשחקים שהקבוצה שיחקה (זאת בעלת teamid1) נשאר כמו שהיה מקודם. לאחר רכישת הקבוצה, כל השחקנים שהצטרפו מקבוצה 2 ייחשבו כשחקנים חדשים יותר מכל השחקנים שהיו בקבוצה לפני האיחוד, כאשר השחקן הכי חדש בקבוצה 2 יהיה השחקן הכי חדש בקבוצה לאחר האיחוד וכן הלאה.

<u>פרמטרים</u>:

teamId1	מזהה הקבוצה הרוכשת.
teamId2	מזהה הקבוצה הנרכשת.

<u>ערך החזרה:</u>

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $\text{teamId1} \le 0, \text{teamId2} \le 0$ או $\text{teamId1} = \text{teamId2}$.
FAILURE	אם אין קבוצה עם מזהה teamId1 או teamId2.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log(n)) + k_{players\ in\ team1} + k_{players\ in\ team2}$ משוערך, כאשר n הוא מספר הקבוצות במערכת, $k_{players\ in\ team1}$ הוא מספר השחקנים בקבוצה עם המזהה teamid1 ו $k_{players\ in\ team2}$ הוא מספר השחקנים בקבוצה עם המזהה teamid2.

---

output_t<int> play_tournament(int lowPower, int highPower)

נערך טורניר בין כל הקבוצות שהכוח שלהן הוא מספר בין lowPower ל highPower (כולל lowPower ו highPower), כאשר אם יש i קבוצות שהכוח שלהן בטווח המבוקש, בסיבוב הראשון לכל $m \le \frac{i}{2}$, הקבוצה ה m החזקה ביותר משחקת נגד הקבוצה ה $m + \frac{i}{2}$ החזקה ביותר והקבוצה המנצחת (כלומר החזקה מבין השתיים) ממשיכה לסיבוב הבא. בסיבוב הבא לכל $m \le \frac{i}{4}$ הקבוצה ה m החזקה ביותר מבין הקבוצות הנותרות משחקת נגד הקבוצה ה $m + \frac{i}{4}$ החזקה ביותר מבין הקבוצות הנותרות וכן הלאה, עד שנשארות רק שתי קבוצות שמשחקות ביניהן והחזקה מביניהן תנצח. לכל קבוצה שניצחה במשחק כלשהו בטורניר מספר הנצחונות יגדל במספר המשחקים בו היא ניצחה. הטורניר יכול להתקיים רק אם i הוא חזקה של 2.

לדוגמא אם lowPower =4 ו highPower =11 והקבוצות במערכת הן בעלות הכוחות (1,3,4,5,6,7,8,9,10,13) אז הקבוצות המשתתפות יהיו אלה עם הכוחות: (4,5,6,6,7,8,9,10), אלה 8 קבוצות לכן הטורניר יכול להתקיים. כעת הקבוצה בעלת הכוח 10 תשחק נגד הקבוצה בעלת הכוח 6, הקבוצה בעלת הכוח 9 תשחק נגד הקבוצה השנייה בעלת הכוח 6, הקבוצה בעלת הכוח 8 תשחק נגד הקבוצה בעלת הכוח 5 והקבוצה בעלת הכוח 7 תשחק נגד הקבוצה בעלת הכוח 4. הקבוצות בעלות הכוחות (7,8,9,10) ינצחו ויעברו לסיבוב הבא וזה ימשך באותו אופן.

<u>פרמטרים</u>:

lowPower	ערך הכוח המינימלי שמאפשר השתתפות בטורניר.
highPower	ערך הכוח המקסימלי שמאפשר השתתפות בטורניר.

<u>ערך החזרה:</u> מזהה הקבוצה המנצחת בטורניר. ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $lowPower \le 0$ או $highPower \le 0$ או $highPower \le lowPower$.
FAILURE	אם מספר הקבוצות בטווח הנתון הוא לא חזקה של 2.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log(i) \cdot \log(n))$ במקרה הגרוע, כאשר n הוא מספר הקבוצות במערכת ו i הוא מספר הקבוצות המשתתפות בטורניר.

<u>סיבוכיות מקום</u>:

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא $O(n+k)$ במקרה הגרוע, כאשר n הוא מספר הקבוצות ו-k הוא מספר השחקנים. כלומר בכל רגע בזמן הריצה, צריכת המקום של מבנה הנתונים תהיה לינארית בסכום מספרי השחקנים 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:</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:

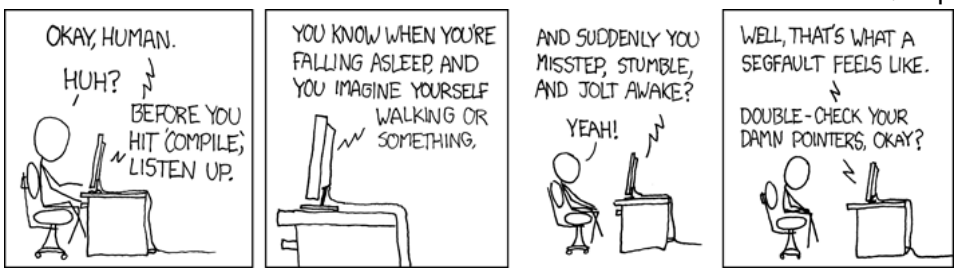
- החלק היבש הווה חלק מהציון על התרגיל כפי שמצוין בנהלי הקורס.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כלומר, הגשה בה יתקבל אך ורק חלק יבש תגרור ציון 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 olympics24a2.h.
- קראו היטב את הקובץ הנ"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים main24a2.cpp ו-wet2util.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 main24a2.cpp ו-wet2util.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.
- **<u>שימו לב:</u>** 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.

<u>הגשה:</u>

- **<u>חלק יבש+ חלק רטוב</u>**:

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

- בתיקייה הראשית:

קבצי ה-Source Files שלכם. **למעט הקבצים main24a2.cpp ו-wet2util.h**, שאסור לשנות.
קובץ PDFבשם dry.pdfאשר 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

- **<u>שימו לב כי אתם מגישים את כל שלושת החלקים הנ"ל.</u>**
- 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).

<u>דחיות ואיחורים בהגשה:</u>

- 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 האחראיבלבד 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!