מתרגל ממונה על התרגיל: תומר אדר, tomer-adar@cs.technnion.ac.il

תאריך ושעת הגשה: 05/01/2023 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/winter2023/234218
  - 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 תחילת המשחקים בגביע העולם בכדורגל (קטאר 2022), האנליסטים רוצים לעקוב אחרי מהלכים עסקיים "מאחורי הקלעים". הם מבקשים את עזרתכם בבניית מבנה הנתונים שיהווה את הבסיס ל-backend של האפליקציה.

כל שחקן או קבוצה במערכת מיוצגים על ידי מזהה מספרי ייחודי. מכיוון שהטורניר כבר התחיל, החוקים שונים מאלו שנדרשו למבנה הנתונים שנדרש מכם לפני חודש.

חוקי הטורניר

1. שחקן לא יכול לעזוב קבוצה.
2. אם קבוצה מודחת מהטורניר, כל השחקנים בקבוצה עוזבים את הטורניר ולא חוזרים אליו יותר, אבל המידע שלהם נשאר במערכת.
3. השחקנים בקבוצה מסודרים לפי סדר ההצטרפות הכרונולוגי שלהם.
4. אם קבוצה אחת קונה קבוצה אחרת - כל השחקנים שומרים על הסדר המקורי בקבוצה שלהם, אבל כל השחקנים של הקבוצה הרוכשת נחשבים לוותיקים יותר מכל השחקנים של הקבוצה הנרכשת. באופן מפורש:

$$A = (a_1, \dots, a_{|A|}), \qquad B = (b_1, \dots, b_{|B|}) \quad \Rightarrow \quad A \text{ buys } B \text{ yields } (a_1, \dots, a_{|A|}, b_1, \dots, b_{|B|})$$

5. לכל שחקן יש "רוח", spirit, שמתוארת על ידי פרמוטציה מהקבוצה {1,2,3,4,5} לעצמה.
6. לכל קבוצה יש "רוח קבוצתית", team spirit שהיא גם פרמוטציה שמתקבלת מהרכבה ("כפל") של כל רוחות השחקנים שלה לפי הסדר באופן המפורש הבא:

$$A = (a_1, \dots, a_{|A|}), \quad \text{team-spirit}(A) = \prod_{i=1}^{|A|} \text{player-spirit}(a_i)$$

7. באחת הפונקציות תידרשו להשוות "כוח רוחני" (team strength) של שתי קבוצות. החישוב של "כוח רוחני" בהינתן "רוח קבוצתית" (team spirit) הוא קופסה שחורה. (ניתן לראות את מימוש הפונקציה בקובץ wet2Util.h)

נקודות חשובות

1. סיפקנו לכם את המחלקה permutation_t עם פעולות יצירה, הרכבה ("כפל"), היפוך, בדיקת תקינות וחישוב "כוח רוחני". אין צורך לממש את המחלקה הזו בעצמכם.
2. בכל אחת מהפונקציות שזמן הריצה שלה משוערך פרט לפונקציה add_player, השערוך הוא יחד עם כל הפונקציות האחרות פרט לפונקציה add_player.

באופן מפורש, אם הפונקציות $f_1, \dots, f_n$ נדרשות לזמן ריצה משוערך של $t_1, \dots, t_n$ בהתאמה, ומספר הפעמים שקוראים לכל אחת מהן הוא $c_1, \dots, c_n$ בהתאמה, נדרש שזמן הריצה הכולל של הקריאות האלו יהיה:

$$O(c_1 t_1 + c_2 t_2 + \cdots + c_n t_n)$$

3. המשמעות של "משוערך, בממוצע על הקלט" היא שעבור קלט ממוצע, זמן הריצה המשוערך יהיה כפי שצוין.

דרוש מבנה נתונים למימוש הפעולות הבאות:

world_cup_t()

מאתחלת מבנה נתונים ריק. תחילה אין במערכת קבוצות או שחקנים.

פרמטרים: אין.

ערך החזרה: אין.

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

---

virtual ~world_cup_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 השחקנים הכולל ו-K הוא מספר הקבוצות מאז תחילת המערכת.

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 $\text{teamId} \le 0$.
FAILURE	אם teamId הוא מזהה של קבוצה קיימת.
SUCCESS	במקרה של הצלחה.

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

StatusType remove_team(int teamId)

הקבוצה בעלת המזהה teamId מודחת מהטורניר, ולכן צריך להוציאה מהמערכת, יחד עם כל שחקניה, שכבר לא ישתתפו בטורניר תחת קבוצה אחרת.

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 $\text{teamId} \le 0$.
FAILURE	אם אין קבוצה בעלת מזהה teamId.
SUCCESS	במקרה של הצלחה.

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

StatusType add_player(int playerId, int teamId, const permutation_t &spirit, int gamesPlayed, int ability, int cards, bool goalKeeper)

השחקן בעל מזהה ייחודי playerId משחק עבור הקבוצה בעלת המזהה teamId בתחרות. ה"רוח" שלו נתונה על ידי spirit. נתון שהשחקן שיחק gamesPlayed משחקים ושהמסוגלות שלו היא ability. Cards הוא מספר הכרטיסים ההתחלתי של השחקן, ו- goalKeeper מציין האם הוא שוער או לא.

פרמטרים:

playerId	מזהה השחקן שצריך להוסיף.

teamId	מזהה הקבוצה של השחקן.
spirit	אובייקט שמתאר את האופי של השחקן.
gamesPlayed	מספר המשחקים שהשחקן שיחק עד כה.
ability	היכולת של השחקן (יכולה להיות גם שלילית או אפס).
cards	מספר הכרטיסים ההתחלתי של השחקן.
goalKeeper	האם השחקן הוא שוער.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $playerId \leq 0$ או $teamId \leq 0$ או אם ה-spirit לא תקין או $gamesPlayed < 0$ או אם $cards < 0$.
FAILURE	אם קיים כבר שחקן עם מזהה playerId, או שהיה קיים בעבר שחקן עם מזהה playerId, או שהקבוצה עם המזהה teamId לא קיימת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log k)$ משוערך, בממוצע על הקלט, כאשר k הוא מספר הקבוצות במערכת.

---

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

שתי הקבוצות בעלות המזהים teamId1 ו-teamId2 משחקות אחת מול השנייה בטורניר.
קבוצות יכולות לשחק רק אם לכל אחת מהן יש לפחות שוער אחד.
(שימו לב שבניגוד לתרגיל הקודם, אין אילוץ על מספר השחקנים הכולל).
היכולת של קבוצה לנצח מוגדרת לפי:

$$\text{points} + \sum_{\text{players}} \text{player ability}$$

אם שני הערכים שונים: הקבוצה בעלת היכולת הגדולה יותר תנצח.
אם שני הערכים שווים: מחשבים את רוח הקבוצה (team spirit) ואת ה"כוח הרוחני" שלה (spirit strength).
הקבוצה בעלת הכוח הרוחני הגדול יותר תנצח, ורק אם לשתי הקבוצות יש כוח רוחני בעוצמה זהה המשחק יסתיים בתיקו.
כרגיל, ניצחון שווה +3 נקודות לקבוצה המנצחת ותיקו שווה נקודה בודדת לכל אחת מהקבוצות.
מספר המשחקים שכל אחד מהשחקנים בשתי הקבוצות שיחק (gamesPlayed) גדל ב-1 עבור כל השחקנים.

פרמטרים:

teamId1	מזהה הקבוצה הראשונה.
teamId2	מזהה הקבוצה השנייה.

ערך החזרה: 0 אם היה תיקו; 1 אם הקבוצה הראשונה ניצחה לפי יכולת; 2 אם הקבוצה הראשונה ניצחה לפי רוח; 3 אם הקבוצה השניה ניצחה לפי יכולת; 4 אם הקבוצה השניה ניצחה לפי רוח. ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $teamId1 \leq 0$, $teamId2 \leq 0$ או $teamId1 = teamId2$.
FAILURE	אם אין קבוצה עם מזהה teamId1 או teamId2 או אם לפחות באחת מהן אין שחקן שהוא שוער.
SUCCESS	במקרה של הצלחה.

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

---

output_t<int> num_played_games_for_player(int playerId)

יש להחזיר את מספר המשחקים הכולל שהשחקן בעל מזהה playerId השתתף בהם, בין אם השחקן עדיין פעיל ובין אם הוא הודח מהתחרות.

פרמטרים:

playerId מזהה השחקן.

ערך החזרה: מספר המשחקים הכולל בהם השתתף השחקן, ובנוסף סטטוס:

ALLOCATION_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT אם $playerId \leq 0$.

FAILURE אם מעולם לא היה שחקן עם מזהה playerId.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* n)$ משוערך, בממוצע על הקלט, כאשר n מספר השחקנים במערכת (כולל שחקני עבר).

StatusType add_player_cards(int playerId, int cards)

השחקן בעל מזהה playerId קיבל cards כרטיסים.

פרמטרים:

playerId מזהה השחקן.

ערך החזרה:

ALLOCATION_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT אם $playerId \leq 0$ או אם $cards < 0$.

FAILURE אם מעולם לא היה שחקן עם מזהה playerId או אם הוא הודח מהטורניר.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* n)$ משוערך, בממוצע על הקלט, כאשר n מספר השחקנים במערכת (כולל שחקני עבר).

---

output_t<int> get_player_cards(int playerId)

יש להחזיר את מספר הכרטיסים הכולל שהשחקן בעל מזהה playerId קיבל, בין אם השחקן עדיין פעיל ובין אם הוא הודח מהתחרות.

פרמטרים:

playerId מזהה השחקן.

ערך החזרה: מספר הכרטיסים הכולל שהשחקן בעל מזהה playerId קיבל, ובנוסף סטטוס:

ALLOCATION_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID_INPUT אם $playerId \leq 0$.

FAILURE אם מעולם לא היה שחקן עם מזהה playerId.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ בממוצע על הקלט.

---

output_t<int> get_team_points(int teamId)

יש להחזיר את מספר הנקודות של הקבוצה בעלת המזהה teamId.

פרמטרים:

teamId	מזהה הקבוצה.

ערך החזרה: מספר הנקודות של הקבוצה teamId, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם teamId ≤ 0.
FAILURE	אם אין קבוצה במערכת עם מזהה teamId.
SUCCESS	במקרה של הצלחה.

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

---

output_t<int> get_ith_pointless_ability(int i)

נניח שמסדרים את הקבוצות לפי היכולת שלהן לנצח כאשר מתעלמים ממספר הנקודות שלהן.
כלומר לפי:

$$\sum_{\text{players}} \text{player ability}$$

שובר שוויון עבור קבוצות עם אותה יכולת יהיה לפי teamId בסדר עולה.
יש להחזיר את ה-id של הקבוצה במיקום ה-i לפי הסדר הזה. (אינדקס ראשון = 0).

פרמטרים:

i -	אינדקס הקבוצה המבוקשת לפי הסדר שהוגדר.

ערך החזרה: מזהה של הקב, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
FAILURE	אם אין קבוצות במערכת או אם i < 0 או i ≥ number-of-teams.
SUCCESS	במקרה של הצלחה.

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

---

output_t<permutation_t> get_partial_spirit(int playerId)

רוצים לחשב את ה"רוח" של הקבוצה שבה חבר השחקן עם מזהה playerId באופן חלקי בלבד. השחקנים שנלקחים בחשבון בחישוב הם השחקן בעל המזהה playerId והשחקנים שהצטרפו לפניו (לפי הסדר הכרונולוגי), ולא כל השחקנים בקבוצה שלו.
אם השחקן בעל המזהה playerId הוא השחקן ה-i לפי הסדר הכרונולוגי של הקבוצה שלו, יש לחשב את:

$$\prod_{j=1}^{i} \text{spirit(the } j\text{th player in the team of } playerId)$$

שימו לב: יש להחזיר את הפרמוטציה עצמה שמתקבלת מההרכבה ("כפל").

פרמטרים:

playerId	מזהה השחקן.

ערך החזרה: החישוב המבוקש, ובנוסף סטטוס:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם playerId ≤ 0.
FAILURE	אם מעולם לא היה שחקן עם מזהה playerId או שהוא הודח מהטורניר.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* n)$ משוערך, בממוצע על הקלט.

StatusType buy_team(int buyerId, int boughtId)

הקבוצה בעלת מזהה buyerId קונה את הקבוצה בעלת מזהה boughtId. המזהה של הקבוצה המאוחדת נשאר זהה למזהה של הקבוצה הרוכשת.

הניקוד של הקבוצה הממוזגת יהיה סכום הניקוד של הקבוצות המקוריות.

כל שחקן בקבוצה הממוזגת, מספר המשחקים ששיחקן נשמר.

<u>פרמטרים</u>:

buyerId	מזהה קבוצה ראשונה.
boughtId	מזהה קבוצה שניה.

<u>ערך החזרה</u>:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם אחד ה-id-ים אינו חיובי או שהם שווים.
FAILURE	אם אין קבוצות עם ה-id הנתונים.
SUCCESS	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log k + \log^* n)$ משוערך, כאשר k הוא מספר הקבוצות ו-n מספר השחקנים במערכת (כולל שחקני עבר).

<u>סיבוכיות מקום</u>:

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא $O(n+k)$ במקרה הגרוע, כאשר n הוא מספר השחקנים ו-k הוא מספר הקבוצות. 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:

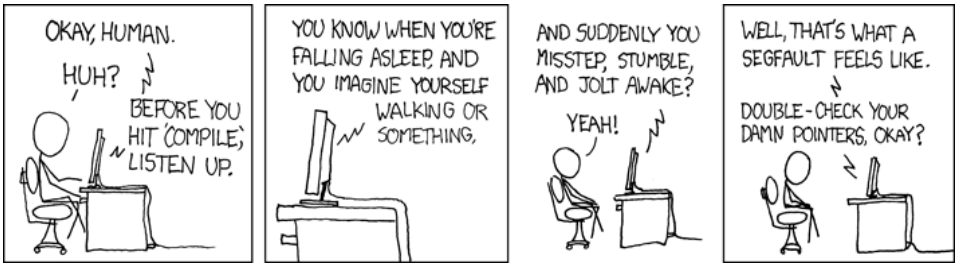
- החלק היבש הווה חלק מהציון על התרגיל כפי שמצוין בנהלי הקורס.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כלומר, הגשה בה יתקבל אך ורק חלק יבש תגרור ציון 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 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.
- **<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 main23a1.cpp ו-wet1util.h**, שאסור לשנות.
  - קובץ PDFבשם 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

- **<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!