

# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 2 – מעודכן לתאריך 04.01.2026  
עמוד 1 מתוך 9



מתרגל ממונה על התרגיל: יצחק גרינבוים, [Yitzchakg@campus.technion.ac.il](mailto:Yitzchakg@campus.technion.ac.il)

תאריך ושעת הגשה: 21/01/2026 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/technion.ac.il/winter2026/234218>, 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 כהודעות 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 פתרוןכם, 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".
- ההמלצות לתכנות במסמך זה אינן מחייבות, אך מומלץ להיעזר בהן.
- חומר התרגיל הינו כל החומר שנלמד בהרצאות ובתרגולים עד מיונים (לא כולל).
  - 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: [goldstein@campus.technion.ac.il](mailto:goldstein@campus.technion.ac.il).
- בהצלחה!



## הקדמה:

"יצורים נדירים.

מפלצות אימתניות.

אוצרות יקרי ערך.

ממלכות שטניות.

עולמות נסתרים.

הצירוף 'לא-נודע' מכיל מעין קסם משלו. ישנם אלה שנשבו בקסם הזה.

קוראים להם **האנטרים** (ציידים)"

האנטרים הם צייד-עילית בעלי רישיון מטעם אגודת ההאנטרים.

יכולותיהם הפיזיות, המנטליות ובעיקר שליטתם ב**נן** – אנרגיית חיים הניתנת לתיעול

לחיזוק הגוף, מניפולציה של חפצים, חישה מרחוק ויכולות התקפיות מיוחדות, מאפשרות להם לבצע משימות

שלא כל אדם מסוגל להן, כמו חקר אזורים מסוכנים, ציד יצורים נדירים, ליווי, הגנה ופתרון סכסוכים מסוכנים.

עם זאת, לא כל האנטר הוא גיבור. נן יכול לשמש לטוב או לרע. יש הפועלים למען האנושות, ויש

הרודפים אחר אינטרסים אישיים, עושר או כוח, ולעיתים עד כדי עימותים עקובי דם בין האנטרים

שונים. לעתים קרובות האנטרים מתאגדים לצוותים שפועלים יחד.

יושב ראש ה-12 של הארגון, אייזק נטרו, ששמע על יכולותיכם כמתכנתי עילית ולא מכיר את

**chatgpt** כל כך טוב, מעוניין להעביר את המערכת הישנה למעקב אחר פעולות ההאנטרים

למערכת חכמה חדשה בשם '**Huntech**', והוא מבקש את עזרתכם בפיתוחה.

כל האנטר או צוות במערכת יהיו מיוצגים על ידי מספר מזהה ייחודי.



## חוקים והנחות של מערכת ההאנטרים

1. האנטר לא עוזב צוות שהוא פעיל בו.

2. כשצוות מאבד את חבריו בקרב, הצוות הופך ללא פעיל, אבל המידע על ההאנטרים נשאר במערכת.

3. האנטרים בצוות מסודרים לפי **הסדר הכרונולוגי** בו הצטרפו.

4. לכל האנטר יש **יכולת נן** (NenAbility), שהיא אחת מבין 6 אפשרויות:

משפר (Enhancer), משנה (Transmuter), מכשף (Conjurer), פולט (Emitter), מתמך (Manipulator),

מומחה (Specialist). ובנוסף **הילה** (Aura), שלם אי שלילי המתאר את כמות הנן שיש להאנטר.

5. לכל צוות **יכולת נן משותפת**, שהיא סכום יכולות הנן של חבריה (הסבר בנקודות למטה), וגם **ניסיון** (מספר שלם).

6. צוות האנטרים יכול לכפות על צוות אחר להצטרף אליו במידה ותנאים מסוימים מתקיימים (מפורט היכן שרלוונטי).

במקרה כזה, כל ההאנטרים שהיו בצוות הכופה ייחשבו קודמים בסידור הכרונולוגי של ההצטרפות לצוות. במפורש,

אם  $A$  כופה את  $B$  להצטרף אליו:

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

## נקודות חשובות

- סיפקנו לכם את המחלקה NenAbility, עם פעולות יצירה, העתקה, השמה, בדיקת תקינות, איבר אפס,

פעולות אריתמטיות (+, -, \*, /, =), שלילה, אופרטור השוואה (<, >, ==), יכולת נן אפקטיבית והדפסה. אין

צורך לממש את המחלקה הזו בעצמכם.

- בכל אחת מהפונקציות שזמן הריצה שלה משוערך פרט לפונקציה add\_hunter, השערוך הוא יחד עם כל

הפונקציות האחרות פרט לפונקציה add\_hunter.



### דרוש מבנה נתונים למימוש הפעולות הבאות:

Huntech()

מתחלת מבנה נתונים ריק. תחילה אין במערכת צוותים או האנטרים.

פרמטרים: אין.

ערך החזרה: אין.

סיבוכיות זמן:  $O(1)$  במקרה הגרוע.

virtual ~Huntech ()

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

פרמטרים: אין.

ערך החזרה: אין.

סיבוכיות זמן:  $O(n + k)$  במקרה הגרוע, כאשר  $n$  הוא מספר ההאנטרים הכולל במערכת ו- $k$  הוא מספר הצוותים מאז תחילת המערכת.

StatusType add\_squad(int squadId)

הצוות בעל המזהה squadId נרשם באגודת ההאנטרים, לכן יש לצרפו למערכת. לצוות אין ניסיון בכלל. בעת ההכנסה אין האנטרים בצוות.

פרמטרים:

squadId מזהה הצוות החדש.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $squadId \leq 0$ .
FAILURE	אם squadId הוא מזהה של צוות פעיל קיים.
SUCCESS	במקרה של הצלחה.

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

StatusType remove\_squad(int squadId)

חברי הצוות בעל המזהה squadId נהרגו בקרב איום, ולכן צריך להוציאו מהמערכת, יחד עם כל חבריו. לאחר המחיקה, יתכן שיתווסף למבנה צוות אחר בעל אותו מזהה.

פרמטרים:

squadId מזהה הצוות.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $squadId \leq 0$ .
FAILURE	אם אין צוות בעל מזהה squadId.
SUCCESS	במקרה של הצלחה.

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



StatusType add\_hunter (int hunterId, int squadId, const NenAbility &nenAbility, int aura, int fightsHad):

האנטר בעל המזהה hunterId מצטרף לצוות בעל המזהה squadId. יכולת הנ שלו היא nenAbility, וההילה שלו היא aura. בנוסף נתון שההאנטר השתתף ב-fightsHad קרבות.

# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 2 – מעודכן לתאריך 04.01.2026  
עמוד 4 מתוך 9



## פרמטרים:

מזהה ההאנטר שצריך להוסיף.	hunterId
מזהה הצוות של ההאנטר.	squadId
אובייקט שמתאר את יכולת הנ של ההאנטר.	nenAbility
מספר שלם המתאר את כמות הנ של ההאנטר.	aura
מספר הקרבות בהם השתתף ההאנטר.	fightHad

## ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $hunterId \leq 0$ או $squadId \leq 0$ או $nenAbility$ לא תקין (wet2util.h)	INVALID_INPUT
או $aura < 0$ או $fightHad < 0$ .	
אם קיים כבר האנטר עם מזהה $hunterId$ , או שהיה קיים בעבר האנטר עם מזהה $hunterId$ , או שהצוות עם המזהה $squadId$ לא קיים במערכת.	FAILURE
במקרה של הצלחה.	SUCCESS

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

`output_t<int> squad_duel(int squadId1, int squadId2)`

שני צוותי ההאנטרים בעלי המזהים  $squadId1$  ו- $squadId2$  הסתכסכו והם נלחמים. נסמן את הניסיון של הצוות  $squadExp$ . הילת הקרב האפקטיבית של צוות מוגדרת לפי:

$$squadExp + \sum_{hunters} hunter\ aura$$

אם לצוות אחד הילת קרב אפקטיבית גדולה יותר: הצוות בעל הערך הגדול יותר ינצח. אם שני הערכים שווים: משווים את יכולות הנ של הצוותים. הצוות בעל יתרון ביכולת נ ינצח (השתמשו באופרטור ההשוואה הלא טרנזיטיבי של  $NenAbility$ ), ורק אם לשני הצוותים יש יכולת נ זהה הקרב יסתיים בתיקו. **ניצחון** מגדיל את  $squadExp$  של הצוות המנצח ב-3, **ותיקו** מגדיל ערך זה ב-1 לשני הצוותים.

מספר הקרבות שכל אחד מההאנטרים משני הצוותים השתתף בהם גדל ב-1.

## פרמטרים:

מזהה הצוות הראשון.	squadId1
מזהה הצוות השני.	squadId2

**ערך החזרה:** 0 בתיקו; 1 אם הצוות הראשון ניצח בגלל הילת קרב אפקטיבית; 2 אם הצוות הראשון ניצח בגלל יתרון ביכולת נ; 3 אם הצוות השני ניצח בגלל הילת קרב אפקטיבית; 4 אם הצוות השני ניצח בגלל יתרון ביכולת נ. ובנוסף סטטוס:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $squadId1 \leq 0$ , $squadId2 \leq 0$ או $squadId1 = squadId2$ .	INVALID_INPUT
אם אין צוות פעיל עם מזהה $squadId1$ או $squadId2$ , <b>או אחד הצוותים ריק</b> .	FAILURE
במקרה של הצלחה.	SUCCESS

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

`output_t<int> get_hunter_fights_number(int hunterId)`

יש להחזיר את מספר הקרבות הכולל שההאנטר בעל המזהה  $hunterId$  השתתף בהם, בין אם הוא חי ובין אם נהרג.

## פרמטרים:

מזהה ההאנטר.	hunterId
<b>ערך החזרה:</b> מספר הקרבות הכולל בהם השתתף ההאנטר, ובנוסף סטטוס:	
במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $hunterId \leq 0$ .	INVALID_INPUT
אם מעולם לא היה האנטר עם מזהה $hunterId$ .	FAILURE
במקרה של הצלחה.	SUCCESS

סיבוכיות זמן:  $O(\log^* n)$  משוערך, בממוצע על הקלט, כאשר  $n$  מספר ההאנטרים שהוכנסו למערכת מעולם.

`output_t<int> get_squad_experience(int squadId)`

יש להחזיר את הניסיון של צוות ההאנטרים בעל המזהה `squadId`.

פרמטרים:

<code>squadId</code>	מזהה הצוות.
<u>ערך החזרה:</u> הניסיון של הצוות <code>squadId</code> , ובנוסף סטטוס:	
ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם <code>squadId ≤ 0</code> .
FAILURE	אם אין צוות במערכת עם מזהה <code>squadId</code> .
SUCCESS	במקרה של הצלחה.
<u>סיבוכיות זמן:</u> $O(\log k)$ במקרה הגרוע, כאשר $k$ מספר הצוותים במערכת.	

`output_t<int> get_ith_collective_aura_squad(int i)`

נניח שמסדרים את צוותי ההאנטרים במערכת לפי ההילה המשותפת שלהם, כלומר לפי:

$$\sum_{\text{hunters}} \text{hunter aura}$$

שובר שוויון עבור צוותים עם אותה הילה משותפת יהיה לפי `squadId` בסדר עולה. יש להחזיר את ה-id של הצוות במיקום ה- $i$  לפי הסדר הזה (אינדקס ראשון = 1).

פרמטרים:

$i$	אינדקס הצוות המבוקש לפי הסדר שהוגדר.
<u>ערך החזרה:</u> מזהה של הצוות, ובנוסף סטטוס:	
ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
FAILURE	אם אין צוותים במערכת או אם $i < 1$ או $i > \text{number-of-squads}$ .
SUCCESS	במקרה של הצלחה.
<u>סיבוכיות זמן:</u> $O(\log k)$ במקרה הגרוע, כאשר $k$ הוא מספר הצוותים במערכת.	

`output_t<NenAbility> get_partial_nen_ability(int hunterId)`

רוצים לחשב את יכולת הנן המשותפת של הצוות בו חבר ההאנטר עם המזהה `hunterId` באופן חלקי בלבד. ההאנטרים שגלגלים בחשבון בחישוב הם ההאנטר בעל המזהה `hunterId` וההאנטרים שהצטרפו לפניו (לפי הסדר הכרונולוגי), ולא כל ההאנטרים בצוות שלו. אם ההאנטר בעל המזהה `hunterId` הוא ההאנטר ה- $i$  לפי הסדר הכרונולוגי של הצוות שלו, יש לחשב את:

$$\sum_{j=1}^i \text{nenAbility of hunter } j \text{ in the squad of hunterId}$$

שימו לב: יש להחזיר את אובייקט יכולת הנן עצמו שמתקבל מהסכימה.

פרמטרים:

<code>hunterId</code>	מזהה ההאנטר.
<u>ערך החזרה:</u> יכולת הנן המבוקשת, ובנוסף סטטוס:	
ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם <code>hunterId ≤ 0</code> .
FAILURE	אם מעולם לא היה האנטר עם מזהה <code>hunterId</code> או שהוא נהרג.
SUCCESS	במקרה של הצלחה.
<u>סיבוכיות זמן:</u> $O(\log^* n)$ משוערך, בממוצע על הקלט.	



`StatusType force_join(int forcingSquadId, int forcedSquadId)`

הצוות בעל מזהה `forcingSquadId` רוצה לכפות על הצוות בעל מזהה `forcedSquadId` להצטרף אליו. צוות יצליח לכפות צוות אחר להצטרף אליו אם הסכום של הניסיון, ההילה המשותפת ויכולת הנן האפקטיבית שלו (השתמשו בפונקציה `getEffectiveNenAbility`) גדול ממש משל האחר (נתייחס לצוות ריק כאילו לעולם אינו יכול לכפות, ותמיד יכול להיכפות על ידי צוות שאינו ריק). במפורש, צוות  $A$  יוכל לכפות את  $B$  להצטרף אליו אם:

$$A's \text{ squadExp} + A's \text{ totalAura} + A's \text{ effectiveNenAbility} > B's \text{ squadExp} + B's \text{ totalAura} + B's \text{ effectiveNenAbility}$$

**המזהה** של הצוות הממוזג נשאר זהה למזהה של הצוות הכופה. הניסיון של הצוות הממוזג יהיה סכום של הניסיון של שני הצוותים המקוריים. לכל האנטר בצוות הממוזג, מספר הקרבות בהם השתתף נשמר.

פרמטרים:

מזהה צוות כופה.

`forcingSquadId`

מזהה צוות נכפה.

`forcedSquadId`

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.

`ALLOCATION_ERROR`

אם אחד ה-id-ים אינו חיובי או שהם שווים.

`INVALID_INPUT`

אם אין צוותים עם ה-id הנתונים, או שלא ניתן לבצע את הכפייה כי הצוות

`FAILURE`

הכופה הוא צוות ריק או כי אי השוויון החזק אינו מתקיים.

במקרה של הצלחה.

`SUCCESS`

סיבוכיות זמן:  $O(\log^* n + \log k)$  משוערך, כאשר  $k$  הוא מספר הצוותים במערכת ו- $n$  הוא מספר ההאנטרים שהיו אי פעם במערכת.



input : output

דוגמה הרצה:

```
addSquad 1 : SUCCESS
addSquad 2 : SUCCESS
addSquad 3 : SUCCESS
addHunter 10 1 Enhancer 50 0 : SUCCESS
addHunter 11 1 Conjurer 30 2 : SUCCESS
addHunter 12 2 Emitter 40 1 : SUCCESS
addHunter 13 3 Specialist 10 0 : SUCCESS
squadDuel 1 2 : SUCCESS, 1
getHunterFightsNumber 10 : SUCCESS, 1
getPartialNenAbility 11 : SUCCESS, Enhancer:1, Emitter:0, Transmuter:0, Conjurer:1, Manipulator:0, Specialist:0
getSquadExperience 1 : SUCCESS, 3
getIthCollectiveAuraSquad 1 : SUCCESS, 3
forceJoin 1 3 : SUCCESS //squad 3 is no more
getIthCollectiveAuraSquad 1 : SUCCESS, 2
forceJoin 2 1 : FAILURE
removeSquad 3 : FAILURE
addHunter 15 3 Enhancer 20 0 : FAILURE
addSquad -5 : INVALID_INPUT
squadDuel 1 1 : INVALID_INPUT
```

# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 2 – מעודכן לתאריך 04.01.2026  
עמוד 7 מתוך 9



## סיבוכיות מקום:

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא  $O(n + k)$  במקרה הגרוע, כאשר  $n$  הוא מספר ההאנטרים שנכנסו למערכת מההתחלה ו- $k$  הוא מספר הקבוצות במערכת מההתחלה. כלומר לינארית בכמות פעולות ההוספה. 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 `NenAbility`), 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.



# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 2 – מעודכן לתאריך 04.01.2026

עמוד 8 מתוך 9



## הנחיות ודגשים כלליים:

### חלק יבש:

- החלק היבש הוא 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 תגרוור 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 איננו תיעוד קוד.
- 1. ראשית הציגו את מבני הנתונים בהם השתמשתם. **רצוי ומומלץ להיעזר בציור.**
- 2. 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.
- 3. 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. יש להסביר שינויים או תוספות בפעולות סטנדרטיות של מבנה נתונים שלמדנו אם נעשו כאלה. 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 בכיתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכיתה, 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**, **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 ב-gradescope הינה: `g++ -std=c++14 -DNDEBUG -Wall -o main.out *.cpp`
- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ `Huntech26a2.h`.
- אין לשנות את הקבצים `main26a2.cpp` ו-`wet2util.h` אשר סופקו כחלק מהתרגיל, ואין להגיש אותם. ישנה בדיקה אוטומטית שאין בקוד שימוש ב-STL, ובדיקה זו נופלת אם מגישים גם את `main26a2.cpp`.
- את שאר הקבצים ניתן לשנות, ותוכלו להוסיף קבצים נוספים כרצונכם, ולהגיש אותם.
- העיקר שהקוד שאתם מגישים יתקמפל עם הפקודה לעיל, כאשר מוסיפים לו את שני הקבצים `main26a2.cpp` ו-`wet2util.h`.
- עליכם לממש בעצמכם את כל מבני הנתונים (**אין להשתמש במבנים של STL** ואין להעתיק 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::pair`, `std::vector`, `std::iterator`, או כל אלגוריתם של STL, רשימה מלאה של הספריות להן אסור לעשות include נמצאת בקובץ `dont_include.txt`.
- **ניתן** להשתמש במצביעים חכמים (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, וודאו שאתם משתמשים בהם נכון. (תזכרו שהם לא פתרון קסם, למשל, כאשר יוצרים מעגל בהצבעות). שימו לב שהעתקת מצביעים חכמים היא איטית מאד ולכן יכולה לגרום ל-timeout. עדיף להעביר `shared_ptr` כשרוצים להעביר `by reference`.
- 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 הבדיקות אותן נריך בנתינת הציון, כאשר פרסמנו 5 מתוך 50.



# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 2 – מעודכן לתאריך 04.01.2026

עמוד 9 מתוך 9



- 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. צורף קובץ readme.txt למקרה שלא ברור איך להריץ את הטסטים.
- במידה ויש timeout על אחד מהטסטים זה יחשב ככישלון בטסט. עליכם לכתוב קוד יעיל במידת הסביר. אין לדאוג מכך יותר מידי, הרוב המוחלט של הפתרונות שעומד בסיבוכיות עומד גם בזמני הריצה.
- **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](#).

### החלק הרטוב:

יש להגיש רק את קבצי הקוד שלכם (לרוב קבצי .h, .cpp, בלבד ואפשר גם להגיש אותם כקובץ zip).

### החלק היבש:

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

### שימו לב כי אתם מגישים את שני החלקים הנ"ל, במטלות השונות.

- לאחר ההגשה, יש באפשרותכם לבצע שינויים ולהגיש שוב. ההגשה האחרונה היא זו שתחשב.
- הערכת הציון שמופיעה ב-gradescope אינה הציון 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 ונטפל 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 רק למתרגל האחראי: [goldshstein@campus.technion.ac.il](mailto:goldshstein@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!**