מתרגל ממונה על התרגיל: עידו גליל, idogalil@cs.technion.ac.il

תאריך ושעת הגשה: 07/12/2021 בשעה 23:55

אופן ההגשה: בזוגות. יורד ציון לתרגילים שיוגשו ביחידים בלי אישור מהמתרגל הממונה על התרגיל.

הנחיות כלליות:

- תשובות לשאלות המרכזיות אשר ישאלו יתפרסמו בחוצץ ה FAQ באתר הקורס לטובת כלל הסטודנטים. שימו לב כי תוכן ה **FAQ** הוא מחייב וחובה לקרוא אותו, אם וכאשר הוא יתפרסם. לא יתקבלו דחיות או ערעורים עקב אי קריאת ה FAQ.
- לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסיבוכיות התרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
- 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.
- שאלות על התרגיל יש לפרסם **באתר הפיאצה של הקורס:**
  piazza.com/technion.ac.il/fall2022/234218
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת barakgahtan@cs.technion.ac.il.

הקדמה:

המוסד הטכנולוגי החליט להשיק את "*משחקי התמנון*": משחקים מאתגרים בין סטודנטים שישודרו לקהל **VIP** מצומצם (מצטייני נשיא ודיקן), כשהפרס הנכסף לזוכים הסופיים במשחקים כולל נק"ז רבות בציון גבוה לגליון הציונים. לשם כך שלח המוסד הטכנולוגי סוכנים לגייס שחקנים (סטודנטים) ברכבל ובכרמלית (כאשר כל סוכן מפתה שחקנים פוטנציאלים באמצעות משחק שאם יזכו בו יקבלו קורס מל"ג מעניין עם ממוצע גבוה), ולאחר מכן להביא אותם ליעד סודי לשם תחילת המשחקים. במהלך המשחקים, השחקנים יתחלקו לקבוצות ויתחרו זה בזה על הפרס, ולעיתים "יפסלו".

לשם תחילת המשחקים, נדרשת מערכת מידע יעילה שתעזור לנהל את קבוצות השחקנים ואת מצבן.

דרוש מבנה נתונים למימוש הפעולות הבאות:

void* Init()

מאתחל מבנה נתונים ריק.

פרמטרים: אין.

ערך החזרה: מצביע למבנה נתונים ריק או $NULL$ במקרה של כישלון.

סיבוכיות זמן: O(1) במקרה הגרוע.

StatusType AddGroup(void *DS, int GroupID)

הוספת קבוצת שחקנים חדשה עם המזהה GroupID.

פרמטרים:

DS	מצביע למבנה הנתונים.
GroupID	מזהה הקבוצה שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL או GroupID <=0
FAILURE	אם GroupID קיים.
SUCCESS	במקרה של הצלחה.

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

StatusType AddPlayer(void *DS, int PlayerID, int GroupID, int Level)

הוספת שחקן חדש שעבר את שלב Level במשחק, ומשתייך לקבוצה בעל המזהה GroupID.

פרמטרים:

DS	מצביע למבנה הנתונים.
PlayerID	מזהה השחקן.
GroupID	מזהה הקבוצה.
Level	השלב במשחק של השחקן.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL, PlayerID<=0, GroupID<=0 או Level<0
FAILURE	אם קיים כבר שחקן עם מזהה PlayerID או שהקבוצה עם המזהה GroupID לא קיימת.
SUCCESS	במקרה של הצלחה.

סיבוכיות: O(log n + log k) במקרה הגרוע, כאשר n הוא מספר השחקנים ו-k הוא מספר הקבוצות.

StatusType RemovePlayer(void *DS, int PlayerID)

השחקן בעל המזהה PlayerID "נפסל" מהמשחק, וניתן למחוק אותו מהמערכת

פרמטרים: DS מצביע למבנה הנתונים.

PlayerID מזהה השחקן שיש להסיר מהמערכת.

ערך החזרה: INVALID_INPUT אם DS==NULL או PlayerID <=0.

FAILURE אם אין שחקן עם מזהה PlayerID.

SUCCESS במקרה של הצלחה.

סיבוכיות: O(log n) במקרה הגרוע, כאשר n הוא מספר השחקנים.

StatusType ReplaceGroup(void *DS, int GroupID, int ReplacementID)

הקבוצה בעלת המזהה GroupID חודלת מלהתקיים, והקבוצה הקיימת במערכת בעלת המזהה ReplacementID מקבלת את השחקנים שהיו בקבוצה GroupID (בנוסף לשחקנים שכבר נמצאים בקבוצה)

פרמטרים: DS מצביע למבנה הנתונים.

GroupID מזהה הקבוצה שיש להסיר מהמערכת.

ReplacementID מזהה הקבוצה שתקבל את השחקנים שהשתייכו ל-GroupID

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם DS==NULL או GroupID<=0 או ReplacementID<=0 או GroupID==ReplacementID

FAILURE אם אין קבוצה עם מזהה GroupID או ReplacementID.

SUCCESS במקרה של הצלחה.

סיבוכיות: $O(\log k + n_{group} + n_{replacement})$ במקרה הגרוע, כאשר k הוא מספר הקבוצות במערכת, n_{group} מספר השחקנים של הקבוצה שפורקה ו-$n_{replacement}$ הוא מספר השחקנים של הקבוצה המחליפה.

StatusType IncreaseLevel (void *DS, int PlayerID, int LevelIncrease)

הגדלת השלב במשחק של השחקן בעל המזהה PlayerID ב-LevelIncrease .

פרמטרים: DS מצביע למבנה הנתונים.

PlayerID מזהה השחקן שיש לעדכן.

LevelIncrease כמות שלבי המשחק שיש להוסיף לשחקן.

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם DS==NULL, PlayerID <=0 או LevelIncrease<=0

FAILURE אם אין שחקן עם מזהה PlayerID.

SUCCESS במקרה של הצלחה.

סיבוכיות: O(log n) במקרה הגרוע, כאשר n הוא מספר השחקנים.

StatusType GetHighestLevel(void *DS, int GroupID, int *PlayerID)

יש להחזיר את מזהה השחקן שנמצא בשלב (Level) הגבוה ביותר מבין אלו ששייכים ל-*GroupID*.

- *אם GroupID<0 יש להחזיר את השחקן שנמצא בשלב הגבוה ביותר בכל המערכת, כלומר בין כל הקבוצות.*
- *אם לשני שחקנים יש level זהה, השחקן המוחזר יהיה בעל ה-*PlayerID *הקטן יותר.*
- *אם לא הוספו שחקנים ל-GroupID (או במערכת כולה אם GroupID<0) יש להחזיר 1-* ב-PlayerID. *שימו לב שמקרה זה נחשב כ-SUCCESS.*

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	GroupID	מזהה הקבוצה שעבורה נרצה לקבל את המידע.
	PlayerID	מצביע למשתנה שיעודכן למזהה השחקן בשלב הגבוה ביותר.
<u>ערך החזרה</u>:	INVALID_INPUT	אם DS==NULL, PlayerID==NULL או *GroupID* ==0.
	FAILURE	אם *GroupID*>0 ואין קבוצה עם מזהה *GroupID*.
	SUCCESS	במקרה של הצלחה, כלומר כל מצב אחר.

<u>סיבוכיות</u>: אם *GroupID*<0 אז $O(1)$ במקרה הגרוע.

אחרת, $O(\log k)$ במקרה הגרוע. כאשר k הוא מספר הקבוצות.

StatusType GetAllPlayersByLevel (void *DS, int *GroupID*, int **Players, int *numOfPlayers)

יש להחזיר את כל השחקנים ששייכים לקבוצה בעלת המזהה *GroupID* ממוינים לפי השלב שלהם.

- *אם GroupID<0 יש להחזיר את השחקנים בכל המערכת ממוינים לפי השלב שלהם.*
- *השחקנים יוחזרו ממוינים לפי השלב שלהם בסדר **יורד**, אם לשני שחקנים יש את אותו מספר שלב אז יש למיין אותם בסדר **עולה** לפי PlayerID.*
- *אם אין שחקנים שהוקצו ל-GroupID (או במערכת כולה אם GroupID<0) יש להחזיר NULL* ב-Players *ואפס* ב-numOfPlayers, *שימו לב שמקרה זה נחשב כ-SUCCESS.*

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	GroupID	מזהה הקבוצה שעבורה נרצה לקבל את המידע.
	Players	מצביע למערך שיכיל את כל השחקנים ששייכים לקבוצה.
	numOfPlayers	מצביע למשתנה שיעודכן למספר השחקנים במערך.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם אחד המצביעים שווה ל-NULL או *GroupID*==0.
	FAILURE	אם *GroupID*>0 ואין קבוצה עם מזהה *GroupID*
	SUCCESS	במקרה של הצלחה, כלומר כל מצב אחר.

<u>סיבוכיות</u>: אם *GroupID*<0 אז $O(n)$ במקרה הגרוע, כאשר n הוא מספר השחקנים במערכת.

אחרת, $O(n_{\text{GroupID}} + \log k)$ במקרה הגרוע, כאשר n_{GroupID} הוא מספר השחקנים ששייכים לקבוצה בעלת המזהה GroupID ו-k הוא מספר הקבוצות.

שימו לב שאתם מקצים את המערך בגודל המתאים, כמו כן אתם צריכים להקצות את המערך בעצמכם באמצעות malloc (כי הוא ישוחרר בקוד שניתן לכם באמצעות free).

StatusType GetGroupsHighestLevel (void *DS, int numOfGroups, int **Players)

יש להחזיר עבור כל אחת מ-numOfGroups הקבוצות בעלות המזהה GroupID הכי קטן **שיש בהן לפחות שחקן אחד** את השחקן שנמצא בשלב (level) הגבוה ביותר (אם יש יותר משחקן אחד באותו level מאותה קבוצה, יהיה זה השחקן בעל ה-playerID הקטן יותר). כל השחקנים יוחזרו במערך ממוינים לפי מזהה הקבוצה GroupID **בסדר עולה**.

- *דוגמה: אם קיימות במשחק הקבוצות 1,2,3,4, כאשר בקבוצה 3 אין שחקנים ובכל השאר יש לפחות שחקן אחד והתקבלה הפעולה GetGroupsHighestLevel עם numOfGroups=3, יוחזר מערך בו בתא הראשון השחקן בשלב הכי גבוה בקבוצה 1, בתא השני השחקן בשלב הכי גבוה בקבוצה 2 ובתא השלישי השחקן הכי גבוה בקבוצה 4 (קבוצה 3 ריקה משחקנים ולכן לא הוחזר עבורה שחקן).*

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	Players	מצביע למערך שיכיל את כל השחקנים בשלב הגבוה ביותר מכל קבוצה.
	numOfGroups	כמות הקבוצות הלא ריקות מהן צריך להחזיר את השחקנים בשלב הגבוה ביותר.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם אחד המצביעים שווה ל-NULL או numOfGroups<1.
	FAILURE	אם numOfGroups גדול ממספר הקבוצות הלא ריקות.
	SUCCESS	במקרה של הצלחה, כלומר כל מצב אחר.
<u>סיבוכיות</u>:	$O(numOfGroups + \log k)$	במקרה הגרוע, כאשר $numOfGroups$ הוא הפרמטר לפונקציה (כמות הקבוצות הלא ריקות להן יש להחזיר את השחקנים בשלב הכי גבוה) ו-k הוא מספר הקבוצות.

שימו לב שאתם מקצים את המערך בגודל המתאים, כמו כן אתם צריכים להקצות את המערך בעצמכם באמצעות malloc (כי הוא ישוחרר בקוד שניתן לכם באמצעות free).

void Quit(void **DS)

הפעולה משחררת את המבנה. בסוף השחרור יש להציב ערך NULL ב-DS, אף פעולה לא תקרא לאחר מכן.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
<u>ערך החזרה</u>:	אין.	
<u>סיבוכיות</u>:	$O(n + k)$	במקרה הגרוע, כאשר n הוא מספר השחקנים ו-k הוא מספר הקבוצות.

<u>סיבוכיות מקום</u> - $O(n + k)$ במקרה הגרוע, כאשר n הוא מספר השחקנים ו-k הוא מספר הקבוצות.

<u>ערכי החזרה של הפונקציות:</u>

בכל אחת מהפונקציות, ערך ההחזרה שיוחזר ייקבע לפי הכלל הבא:

- תחילה, יוחזר INVALID_INPUT אם הקלט אינו תקין.
- אם לא הוחזר INVALID_INPUT:
  - בכל שלב בפונקציה, אם קרתה שגיאת הקצאה יש להחזיר ALLOCATION_ERROR.
  - אם קרתה שגיאה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד FAILURE מבלי לשנות את מבנה הנתונים.

- אחרת יוחזר SUCCESS.

הנחיות:

חלק יבש:

- **הציון על החלק היבש הוא 50% מהציון של התרגיל.**
- 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.
- 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 מנת לעשות זאת הגדירו מחלקה, נאמר PlayersManager, וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C ב library1.h, ממשו את library1.cpp באופן הבא:

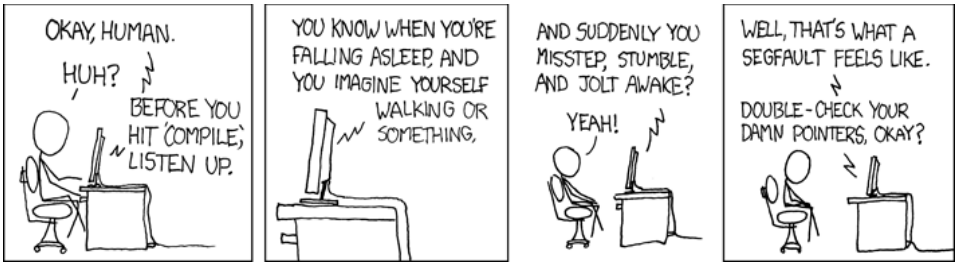
```
#include"library1.h"
#include"PlayersManager.h"

void* Init() {
        PlayersManager *DS = new PlayersManager();
        return (void*)DS;
}
StatusType AddGroup(void *DS, int GroupID){
        return ((PlayersManager*)DS)-> AddGroup (GroupID);
}
```

- 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 library1.h
- קראו היטב את הקובץ הנ"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים אשר סופקו כחלק מהתרגיל, **ואין להגיש אותם**.
- 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.**
- אסור להשתמש ב-std::iterator.
- **ניתן** להשתמש במצביעים חכמים (Smart pointers כמו shared_ptr), בספריית math או בספריית exception.
- מצורפים לתרגיל קבצי קלט ופלט לדוגמא.
- 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 **(ללא הקבצים שפורסמו).**
    - קובץ PDF אשר מכיל את הפתרון היבש עבור. 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:

Roi Bar Zur 012345678 idogalil@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!

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!