מבני נתונים 234218 אביב תשע"ז

גיליון רטוב מספר 1 – מעודכן לתאריך 01.05.2018

מתרגל ממונה על התרגיל: דור זהר, dorzohar@cs.technion.ac.il

תאריך ושעת הגשה: 15.05.2018 בשעה 23:59.

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

הנחיות לפתרון:

- שאלות לגבי דרישות התרגיל יש להפנות באימייל לכתובת הנ"ל.
- 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.**

הקדמה:

ה"אואזיס" הוא עולם של מציאות מדומה, אליו נכנסים שחקנים עם דמויות וירטואליות ("אווטאר"). השחקנים יכולים לחבור לשבטים (**Clans**), להשלים אתגרים ולצבור מטבעות.

עזרו למפתחי האואזיס לנהל את פעולות השחקנים ביעילות

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 addPlayer(void *DS, int playerID, int initialCoins)

הוספת שחקן בעל המזהה playerID ובעל סכום התחלתי של *initialCoins*. בעת ההוספה, השחקן לא נמצא בשבט.

פרמטרים:

DS	מצביע למבנה הנתונים.
playerID	מזהה האווטאר
initialCoins	מספר המטבעות ההתחלתי של השחקן

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם *DS*==NULL, playerID<=0 או *initialCoins*<0
FAILURE	אם קיים כבר שחקן עם מזהה playerID
SUCCESS	במקרה של הצלחה.

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

StatusType addClan(void *DS, int clanID)

הוספת שבט חדש עם המזהה *clanID*.

פרמטרים: *DS* מצביע למבנה הנתונים.

clanID מזהה השבט שיש להוסיף.

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם *DS*==NULL או *clanID*<=0

FAILURE אם *clanID* קיים.

SUCCESS במקרה של הצלחה.

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

StatusType joinClan(void *DS, int player*ID*, int *clanID*)

הוספת השחקן player*ID* לשבט *clanID*. במידה והשחקן *playerID* כבר נמצא בשבט כלשהו, הוא לא יועבר לשבט החדש.

פרמטרים: *DS* מצביע למבנה הנתונים.

player*ID* מזהה השחקן שנכנס לשבט

clanID מזהה השבט אליו נכנס השחקן

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם *DS*==NULL או player*ID*<=0 או *clanID*<=0

FAILURE אם אין שחקן עם מזהה player*ID* או שבט עם מזהה *clanID*, *או שהשחקן playerID* כבר נמצא בשבט כלשהו.

SUCCESS במקרה של הצלחה.

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

StatusType completeChallange(void *DS, int *playerID,* int *coins*)

השחקן בעל המזהה *playerID* השלים אתגר בהצלחה, ולכן קיבל *coins* מטבעות

פרמטרים: *DS* מצביע למבנה הנתונים.

player*ID* מזהה השחקן שהשלים אתגר.

coins מספר המטבעות שיש להוסיף לשחקן

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם *DS*==NULL או player*ID*<=0 או coins<=0

FAILURE אם אין שחקן עם מזהה player*ID*.

SUCCESS במקרה של הצלחה.

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

StatusType getBestPlayer(void *DS, int *clanID*, int *playerID*)

יש להחזיר את מזהה השחקן שהשלים הכי הרבה אתגרים בקבוצה *clanID*

- *אם clanID<0 יש להחזיר את השחקן שהשלים הכי הרבה אתגרים בכל המערכת*
- *אם שני שחקנים השלימו מספר אתגרים מקסימלי, יש להחזיר את זה בעל המזהה הקטן יותר.*
- *אם אין שחקנים בשבט clanID (או במערכת כולה אם clanID<0) יש להחזיר -1 ב-playerID. שימו לב שמקרה זה נחשב כ-SUCCESS.*

פרמטרים:	*DS*	מצביע למבנה הנתונים.
	clanID	מזהה השבט שעבורו נרצה לקבל את המידע.
	playerID	מצביע למשתנה שיעודכן למזהה השחקן שהשלים הכי הרבה אתגרים.
ערך החזרה:	INVALID_INPUT	אם *DS*==NULL, *PlayerID*==NULL או *clanID*==0.
	FAILURE	אם *clanID*>0 ואין שבט בעל מזהה *clanID*
	SUCCESS	במקרה של הצלחה, כלומר כל מצב אחר.

סיבוכיות: אם *clanID*<0 אז $O(1)$ במקרה הגרוע.

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

StatusType getScoreboard(void **DS*, int *clanID*, int ***players*, int **numOfPlayers*)

יש להחזיר את כל השחקנים שנמצאים בשבט *clanID* ממוינים על פי כמות המטבעות שלהם.

- *אם clanID<0 יש להחזיר את השחקנים בכל המערכת ממוינים על סמך כמות המטבעות שלהם.*
- *השחקנים יוחזרו ממוינים לפי כמות המטבעות שלהם בסדר **יורד**, אם לשני שחקנים יש את אותה כמות מטבעות אז יש למיין אותם בסדר **עולה** לפי playerID.*
- *אם אין שחקנים בשבט clanID (או במערכת כולה אם clanID<0) יש להחזיר NULL ב-players ואפס ב-numOfPlayers, שימו לב שמקרה זה נחשב כ-SUCCESS.*

פרמטרים:	*DS*	מצביע למבנה הנתונים.
	clanID	מזהה השבט שעבורו נרצה לקבל את המידע.
	Students	מצביע למערך שיכיל את כל השחקנים שנמצאים בשבט זה
	numOfStudents	מצביע למשתנה שיעודכן למספר השחקנים במערך.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם אחד המצביעים שווה ל-NULL או *clanID*==0.
	FAILURE	אם *clanID*>0 ואין שבט עם מזהה *clanID*
	SUCCESS	במקרה של הצלחה, כלומר כל מצב אחר.

סיבוכיות: אם *clanID*<0 אז $O(n)$ במקרה הגרוע, כאשר n הוא השחקנים במערכת.

אחרת, $O(n_{TeamID} + \log k)$ במקרה הגרוע, כאשר n_{TeamID} הוא מספר השחקנים שנמצאים בשבט *clanID* ו-k הוא מספר השבטים.

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 uniteClans(void **DS*, int *clanID1*, int *clanID2*)

השבטים clanID1 ו-clanID2 החליטו להתאחד, כאשר המזהה של השבט החדש יהיה המזהה של השבט הגדול יותר מבינהם (זה שהכיל יותר שחקנים). אם מספר השחקנים של שני השבטים זהה, יילקח המזהה הנמוך מבין השניים. כל שחקן בשבטים אלו שלא השלים אף אתגר לא ייכנס לשבט החדש (ויישאר ללא שבט).

פרמטרים:	*DS*	מצביע למבנה הנתונים.
	clanID1	השבט הראשון שיש לאחד
	clanID2	השבט השני שיש לאחד
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם *DS*==NULL או *clanID1*<=0 או *clanID2*<=0 או clanID1==clanID2
	FAILURE	אם אין שבט עם המזהה clanID1 או עם המזהה clanID2
	SUCCESS	במקרה של הצלחה.
סיבוכיות:	$O(n + log\ k)$	במקרה הגרוע, כאשר n הוא מספר השחקנים ו-k הוא מספר השבטים.

void Quit(void **DS)

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

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

סיבוכיות מקום - $O(n + k)$ במקרה הגרוע, כאשר n הוא השחקנים ו-k הוא מספר השבטים
.

ערכי החזרה של הפונקציות:

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.

חלק יבש:

- הציון על החלק היבש הוא 50% מציון התרגיל.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- מומלץ לממש תחילה את מבני הנתונים בצורה הכללית ביותר ורק אז לממש את הפונקציות הנדרשות בתרגיל.
- יש להכין מסמך הכולל תיאור של מבני הנתונים והאלגוריתמים בהם השתמשתם בציורף 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.
- 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.
- 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.
- על חלק זה לא לחרוג מ-8 עמודים.
- והכי חשוב **keep it simple!**

חלק רטוב:

- 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 School, 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"Oasis.h"

void* Init() {
        Oasis * DS = new Oasis();
        return (void*)DS;
}
StatusType AddPlayer(void *DS, int PlayerID, int initialCoins){
        return ((Oasis *)DS)->AddPlayer(PlayerID, initialCoins);
}
```

וכולי...

- על הקוד להתקמפל על t2 באופן הבא:
  - **g++ -std=c++0x –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 הנ"ל ואין להגיש אותם.
- עליכם לממש בעצמכם את כל מבני הנתונים (למשל אין להשתמש במבנים של STL ואין להוריד מבני נתונים מהאינטרנט).
- יש לתעד את הקוד בצורה נאותה וסבירה.
- מספר ימים לאחר פרסום התרגיל נפרסם דוגמאות של קבצי קלט וקבצי הפלט המתאימים להם.
- 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 ארוכים, 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** (**ללא תיקיות או תתי תיקיות בתוכו**) שמכיל את הדברים הבאים:

- קבצי ה-Source Files שלכם (ללא הקבצים שפורסמו).
- קובץ PDF אשר מכיל את הפתרון היבש. מומלץ להקליד את החלק הזה. ניתן להגיש קובץ PDF מבוסס על סריקה של פתרון כתוב בכתב יד. שימו לב כי במקרה של כתב לא קריא, כל התרגיל לא ייבדק.
- 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:

Wade Watts 012345678 Parzival@cs.technion.ac.il
Samantha Cook 123456789 Art3mis@cs.technion.ac.il

- **שימו לב כי אתם מגישים את כל שלושת החלקים הנ"ל.**
- אין להשתמש בפורמט כיווץ אחר, מאחר ומערך הבדיקה האוטומטי אינו יודע לזהות פורמטים אחרים.
- אין להגיש קובץ המכיל תתי תיקיות.
- לאחר שהגשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב.
- ההגשה האחרונה היא הנחשבת.
- הגשה שאינה תעמוד בקרטריונים הבאים 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.
- במקרה של איחור מוצדק, יש לצרף לקובץ ה PDF שלכם את סיבות ההגשה באיחור, לפי הטופס המופיע באתר, כולל סריקות של אישורי מילואים או אישורי נבחן.

בהצלחה!