מתרגל ממונה על התרגיל: יאיר פלדמן, yairf11@cs.technion.ac.il

תאריך ושעת הגשה: 17.06.2018 בשעה 23:30

אופן ההגשה: בזוגות.

הנחיות:

- 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.
- בתרגיל זה אין הגבלה על מבני הנתונים בהם אתם יכולים להשתמש. מותר וגם מומלץ להשתמש במבנים שמימשתם בתרגילים הקודמים, אם הם מתאימים לדרישות הסיבוכיות הנוכחיות.
- **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: לאחר שהשחקנים ב"אואזיס" התמקצעו במשחק, הם החליטו שהם רוצים לקחת אותו צעד אחד קדימה ולארגן תחרויות בין שבטים. לצורך הערכת היכולת של השחקנים, לכל שחקן יש ציון שקובע את הרמה שלו לעומת שאר השחקנים.

הפעולות שבהן מבנה הנתונים צריך לתמוך:

```
void*          init(int n, int* clanIDs)
```

מאתחל מערכת חדשה בעלת `n` שבטים. לכל שבט מזהה מייצג שהוא מספר שלם. המזהים של השבטים נתונים ב-`clanIDs`, כמו כן ניתן להניח כי המזהים הנתונים שונים זה מזה. שימו לב כי `n` הוא רק מספר השבטים הנתונים באתחול, אך בהמשך ייתכן ויתווספו עוד שבטים ע"י הפעולה `addClan`.

פרמטרים: `n` מספר השבטים.

`clanIDs` מצביע למערך אשר מחזיק מזהים עבור השבטים.

ערך החזרה: מצביע למבנה נתונים ריק או `NULL` במקרה של כישלון (`n<2`, אם אחד המזהים הוא מספר שלילי, או כל כישלון אחר).

סיבוכיות: $O(n)$ במקרה הגרוע, כאשר n מספר השבטים.

```
StatusType    addClan (void* DS, int clanID)
```

שבט בעל מספר מזהה `clanID` מתווסף למערכת.

פרמטרים:	DS	מצביע למבנה הנתונים.
	clanID	מספרו המזהה של השבט אשר מתווסף למערכת.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL או clanID <0
	FAILURE	אם השבט שמספרו המזהה clanID כבר התווסף למערכת, או במקרה של כל בעיה אחרת.
	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(\log(n))$ בממוצע על הקלט משוערך, כאשר n מספר השבטים במערכת.

```
StatusType   addPlayer(void* DS, int playerID, int score, int clan)
```

שחקן בעל מספר מזהה playerID מתווסף למערכת, ומצטרף לשבט clan.

פרמטרים:	DS	מצביע למבנה הנתונים.
	playerID	מספרו המזהה של השחקן אשר מתווסף למערכת.
	score	הציון של השחקן.
	clan	השבט אליו השחקן מצטרף.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, clan<0, playerID<0 או score<0.
	FAILURE	אם השחקן שמספרו המזהה playerID כבר התווסף למערכת, או שהשבט בעל המזהה clan לא קיים במערכת, או במקרה של כל בעיה אחרת.
	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(\log(m))$ בממוצע על הקלט, כאשר m מספר השחקנים במערכת.

```
StatusType clanFight (void* DS, int clan1, int clan2, int k1, int k2)
```

שני השבטים בעלי המזהים clan2,clan1 נלחמים, כאשר כל שבט מביא את k2,k1 השחקנים בעלי הציונים הגבוהים ביותר שלו בהתאמה. השבט המנצח הוא השבט אשר סכום הציונים של השחקנים שלו הוא הגדול ביותר. במקרה של תיקו, השבט בעל המזהה הנמוך יותר מנצח. לאחר הקרב השבט שניצח כובש את השבט המפסיד, והשבט המפסיד לא יכולה להשתתף יותר בקרבות. עם זאת, עדיין ניתן להוסיף שחקנים לשבט הנכבש והוא עדיין קיים במערכת עם אותם שחקנים בדיוק כמו לפני הקרב.

שימו לב: שחקני השבט הנכבש **לא** עוברים לשבט הכובש. הגדרתו של השבט כנכבש משפיעה אך ורק על אי יכולתו להשתתף בקרבות עתידיים, ולא על שחקני אף אחד מהשבטים. כמו כן, k1, k2 **אינם** נחשבים כקבועים לצורכי סיבוכיות.

פרמטרים:	DS	מצביע למבנה הנתונים.
	clan1	המזהה של השבט הראשון.

	clan2	המזהה של השבט השני.
	k1	מספר השחקנים מהשבט הראשון.
	k2	מספר השחקנים מהשבט השני.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, k1<=0, k1<=0, או clan1/clan2<0.
	FAILURE	אם השבט שמספרו המזהה clan1/clan2 לא נמצא במערכת או שהוא אינו יכולה להשתתף בקרב, או שלשבט clan1/clan2 יש פחות מ-k1/k2 שחקנים (בהתאמה), אם clan1==clan2 או במקרה של כל בעיה אחרת.
	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(log(n) + log(m))$ בממוצע על הקלט, כאשר m מספר השחקנים במערכת, ו-n מספר השבטים במערכת.

```
StatusType    getMinClan (void* DS, int* clan)
```

הפעולה מחזירה את המזהה הקטן ביותר של שבט שעדיין לא נכבש.

פרמטרים:	DS	מצביע למבנה הנתונים.
	clan	מצביע לכתובת בה ישמר המספר של השבט המבוקש.
ערך החזרה:	INVALID_INPUT	אם DS==NULL או clan==NULL.
	FAILURE	במקרה של כל בעיה אחרת.
	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(1)$ במקרה הגרוע.

```
void          quit(void** DS)
```

משחרר את מבנה הנתונים. בסוף השחרור יש להציב ערך NULL ב *DS.

פרמטרים:	DS	מצביע למבנה הנתונים.
ערך החזרה:	אין.	

סיבוכיות: $O(m + n)$ במקרה הגרוע, כאשר n מספר השבטים במערכת ו-m מספר השחקנים במערכת.

סיבוכיות מקום

סיבוכיות המקום של מבנה הנתונים היא $O(m + n)$, כאשרn מספר השבטים במערכת ו-m מספר השחקנים במערכת.

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.
- 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.
- 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.

חלק רטוב:

- אנו ממליצים בחום על מימוש **Object Oriented**, ב**C++**. על מנת לעשות זאת הגדירו מחלקה, נאמר `Oasis`, וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C ב library2.h, ממשו את library2.cpp באופן הבא:

```
#include "library2.h"
#include "Oasis.h"

void* init(int n, int* clansIDs) {
   Oasis* DS = new Oasis(n, clansIDs);
   return (void*)DS;
}

StatusType   getMinClan(void* DS, int* clan) {
   return ((Oasis*)DS)-> getMinClan(clan);
}
```

- 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.

<u>הערות נוספות:</u>

- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ library2.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 (אין להשתמש במבנים של STL ואין להוריד מבני נתונים מהאינטרנט).
- יש לתעד את הקוד בצורה נאותה וסבירה.
- מסופקת לכם דוגמא של קובץ קלט (in.txt) וקובץ הפלט (out.txt) המתאים לו.
- <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 ארוכים, 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** (**ללא תיקיות או תתי תיקיות בתוכו**) שמכיל את הדברים הבאים:
  - קבצי ה-Source Files שלכם **<u>(ללא הקבצים שפורסמו).</u>**
  - 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:

Jane Roe 012345678 janeroe@cs.technion.ac.il

John Doe 123456789 johndoe@cs.technion.ac.il

- **<u>שימו לב כי אתם מגישים את כל שלושת החלקים הנ"ל.</u>**
- 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 שלא תעמוד בקריטריונים הבאים תפסל ותיקנס בנקודות!

העתקות בתוכנה תטופלנה בחומרה! (Buh dum tss)

<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.
- 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!