עבודה מסכמת מבני נתונים
עמוד 1 מתוך 5

הקדמה:

- שאלות לגבי דרישות התרגיל יש להפנות באימייל או Whatsapp.
- לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד **בכל** דרישות הסיבוכיות התרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול

הקדמה:

חברת הסטראט-אפ פרי++ מנסה למכור שירותי מחשוב מתקדמים לפרדסים ברחבי הארץ, במטרה לשפר את הליך הקטיף והמעקב אחרי גידול העצים בפרדס. החברה, ששמעה על כישורי המחשוב המעולים של הסטודנטים בקורס מבני נתונים, החליטה לפנות אליכם לצורך מימוש מערכת המחשוב. תפקיד המערכת יהיה להחזיק מידע עדכני על העצים בפרדס, הפירות הנמצאים עליהם ורמת הבשלות שלהם לקטיפה. עובדי הפרדס יעדכנו את המערכת בהתאם למצב העצים בפועל.המערכת לא מתעניינת בסוג הפרי מתוך ההנחה שכל הפירות בפרדס נתון הינם מאותו הסוג.

להלן הדרישות המדויקות שעל המערכת לקיים:

void* Init(intN)

אתחול פרדס ריק בגודל NxN, כלומר בפרדס יש N שורות, כאשר בכל שורה N גומות לעצים.

פרמטרים: N אורכו ורוחבו של הפרדס.

ערך החזרה: מצביע למבנה נתונים ריק

במקרה שהקצאת הזיכרון נכשלה או ש-N<=0, יוחזר NULL.

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

StatusTypePlantTree(void *DS,int i, int j)

שתילת עץ חדש בגומה ה-(i,j) בפרדס.

סופרים כמובן מאפס ☺

פרמטרים: DS מצביע למבנה הנתונים.

i מספר השורה בה שותלים את העץ

j מספר הגומה בה העץ נשתל בשורה.

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם DS==NULL או i,j לא בתחום[N-1,0] .

FAILURE אם נשתל כבר עץ במקום המדובר.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(log(n))$ במקרה הגרוע, כאשר n הוא מספר העצים **כרגע** במערכת.

שימו לב להבדל בין N– גודל הפרדס, לבין n– מספר העצים כרגע במערכת.

StatusTypeAddFruit(void *DS,int i, intj, intfruitID, int ripeRate)

הוספת פרי עם המזהה *fruitID*, בעל רמת בשלות של *ripeRate*, לעץ שבגומה ה-(i,j) בפרדס.

fruitID הינו מזהה ייחודי של הפרי במערכת, כך שלכל פרי בפדרס fruitID שונה וכל עץ בפרדס מכיל את קבוצת הפירות שהוספנו לו עם ה-fruitIDs המתאימים להם.

פרמטרים: DS מצביע למבנה הנתונים.

i מספר השורה של העץ.

j מספר הגומה של העץ.

fruitID מזהה ייחודי של הפרי.

ripeRate רמת בשלות של הפרי לקטיפה (פירוט בהמשך).

<u>ערך החזרה</u>: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם i,j , DS==NULL לא בתחום[0,0<=fruitID, N-1]

או ripeRate<=0

FAILURE אם קיים כבר פרי עם מזהה fruitID, או שלא קיים עץ בגומה.

SUCCESS במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O((k))$ במקרה הגרוע, כאשר k הוא מספר הפירות בפרדס, ו-n הוא מספר העצים כרגע במערכת.

StatusTypePickFruit(void *DS,intfruitID)

קטיפת הפרי עם המזהה *fruitID*.

<u>פרמטרים</u>: DS מצביע למבנה הנתונים.

fruitID מזהה ייחודי של הפרי

<u>ערך החזרה</u>: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם DS==NULL או fruitID<=0

FAILURE אם לא קיים פרי עם מזהה fruitID

SUCCESS במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O((k))$ במקרה הגרוע, כאשר k הוא מספר הפירות בפרדס, ו-n הוא מספר העצים כרגע במערכת.

StatusTypeRateFruit(void *DS,intfruitID, intripeRate)

עדכון רמת הבשלות של הפרי עם המזהה *fruitID*, לפי הערכת העובד המשקללת את גודל הפרי, צבעו, פגמים שנוצרו בו ו<u>עוד</u>. *דירוג של 1 מתאר פרי מושלם המוכן לקטיפה, וערכים עולים מתארים פירות פחות עדיפים לקטיפה.*

<u>פרמטרים</u>: DS מצביע למבנה הנתונים.

fruitID מזהה ייחודי של הפרי

ripeRate רמת הבשלות של הפרי לקטיפה.

<u>ערך החזרה</u>: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם DS==NULL , fruitID<=0 או ripeRate<=0

FAILURE אם לא קיים פרי עם מזהה fruitID.

SUCCESS במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O((k))$ במקרה הגרוע, כאשר k הוא מספר הפירות בפרדס, ו-n הוא מספר העצים כרגע במערכת.

StatusType GetBestFruit(void *DS, int i, int j, int*fruitID)

החזרת הפרי הטוב ביותר בעץ שבגומה ה-(i,j) בפרדס, *כלומר הפרי בעל ה-ripeRate הקטן ביותר.*

אם לשני פירות יש ripeRate זהה, הפרי הטוב יותר מביניהם יהיה הפרי בעל ה-fruitID הקטן יותר.

אם אין פירות בעץ המתאים יש להחזיר -1 ב-fruitID.

<u>פרמטרים</u>: DS מצביע למבנה הנתונים.

i מספר השורה של העץ

j מספר הגומה של העץ

	fruitID	מצביע למשתנה שיעודכן למזה ההפרי המתאים.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, fruitID ==NULL, או i,j לא בתחום[N-1,0]
	FAILURE	אם לא קיים עץ בגומה.
	SUCCESS	במקרה של הצלחה.

<u>סיבוכיות</u>: $O(log(n))$ במקרה הגרוע, כאשר n הוא מספר העצים כרגע במערכת.

StatusType GetAllFruitsByRate(void *DS,int i, int j, int **fruits, int *numOfFruits)

החזרת כל הפירותבעץ שבגומה ה-(i,j) בפרדסממוינים על סמך רמת הבשלות שלהם.

אתם צריכים להקצות את המערך בעצמכם ***באמצעות malloc (כי זה משוחרר בקוד שניתן לכם באמצעות free).***

אם אין פירות בעץ המתאים יש להחזיר *NULL* ***ב-*** *fruits* ***ואפס ב-*** *numOfFruits***.**

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	i	מספר השורה של העץ
	j	מספר הגומה של העץ
	fruits	מצביע למערך שיכיל את כל הפירות בגומה ממיונים לפי מוכנות בסדר**עולה** (כלומר מתחילים מהבשלים ביותר), אם לשני פירות יש אותה רמת בשלות אז יש למיין אותם בסדר **עולה** לפי fruitID.
	numOfFruits	מצביע למשתנה שיעודכן למספר הפירות במערך.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם אחד מהפרמטרים שווה ל-NULL, או i,j לא בתחום[N-1,0].
	FAILURE	אם לא קיים עץ בגומה.
	SUCCESS	במקרה של הצלחה.

<u>סיבוכיות</u>: $O(k_{tree})$ במקרה הגרוע, כאשר k_{tree} הוא מספר הפירות **בעץ המתאים**, ו-n הוא מספר העצים כרגע במערכת.

שימו לב שאתם מקצים את המערך בגודל המתאים!

StatusTypeUpdateRottenFruits(void *DS,int rottenBase, int rottenFactor)

אחת הבעיות איתן נאלץ הפרדס להתמודד הינה מזיקים שונים אשר תוקפים את פירות הפרדס ובכך פוגעים ברמת הבשלות שלהם לקטיפה. עובדי הפרדס גילו כי את הפרדס תוקפים מזיקים מיוחדים, בוגרי אלגברה, אשר בכל תקיפה פוגעים אך ורק פירות שהמזהה שלהם מתחלק במספר כלשהו, לא ידוע מראש. לשם כך רוצים העובדים פונקציה אשר תאפשר להם לעדכן באופן מיידי את רמת הבשלות של הפירות המותקפים.

עדכון רמת הבשלות של הפירות המותקפים,כל הפירות בפרדס אשר מקיימים $fruitID \% rottenBase == 0$. לכל פרי מותקף יש להכפיל את *רמת הבשלות שלו ב-rottenFactor*.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	rottenBase	המספר על פיו תוקפים המזיקים.

	rottenFactor	פקטור הפגיעה בפירות המותקפים.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL,rottenBase<1 או rottenFactor<1.
	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(n + k)$ במקרה הגרוע, כאשר n הוא מספר העצים ו-k הוא מספר הפירות.

void Quit(void **DS)

הפעולה משחררת את מבנה הנתונים. בסוף השחרור יש להציב ערך NULL ב-DS.

אף פעולה לא תקרא לאחר הקריאה ל Quit.

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

סיבוכיות: $O(n + k)$במקרה הגרוע, כאשר n הוא מספר העצים ו-k הוא מספר הפירות.

סיבוכיות מקום

סיבוכיות המקום של מבנה הנתונים היא$O(n + k)$במקרה הגרוע, כאשר n הוא מספר העצים ו-k הוא מספר הפירות.

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 בה. 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 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 בכיתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכיתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתכוונים.
- והכי חשוב **keep it simple!**

עבודה מסכמת מבני נתונים
עמוד 5 מתוך 5

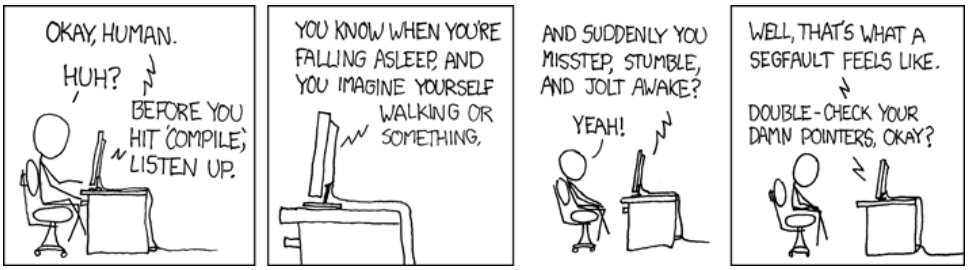
חלק רטוב:

- **מומלץ בחום על להשתמש בC++**, 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 Statistics, 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"Statistics.h"

void* Init() {
	Statistics * DS = new Statistics();
	return (void*)DS;
}
StatusTypePickFruit(void *DS,int fruitID){
	return ((Statistics*)DS)->PickFruit(fruitID);
}
```

וכולי...



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 צורך להגיש אותם.
- יש לתעד את הקוד בצורה נאותה וסבירה.

בהצלחה!