מבני נתונים 234218 חורף תשע"ה
תרגיל רטוב 1 - מעודכן לתאריך 18/11/2014
עמוד 1 מתוך 6

מתרגל ממונה על התרגיל: אלעד ריכרדסון eladrich@cs.technion.ac.il

תאריך ושעת הגשה: 03/12/2014 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

הקדמה:

ענקית המחשוב !Gohoo, העומדת מאחורי מערכת ההפעלה המצליחה iDroid, החליטה לבנות מערכת מחשוב מתקדמת לכבוד השקת הגרסה החדשה של מערכת ההפעלה שלה, iDroid 8. החברה, ששמעה על כישורי המחשוב המעולים של הסטודנטים בקורס מבני נתונים (בפרט בתחום המחשוב הקולינרי), החליטה לפנות אליכם לצורך מימוש המערכת. תפקיד המערכת יהיה להחזיק מידע על כל הגרסאות השונות של מערכת ההפעלה של החברה והאפליקציות השונות שרצות עליה, כאשר כל אפליקציה מחזיקה מידע על מספר ההורדות שלה.

שימו לב: כל אפליקציה מותאמת לגרסה אחת בלבד של מערכת ההפעלה, כך שאם אפליקציה מותאמת למערכת iDroid 8 היא לא תרוץ על מערכת iDroid 10 או iDroid XP.

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 AddVersion(void *DS, int versionCode)

הוספת גרסה חדשה של מערכת ההפעלה עם המזהה *versionCode*, הגרסה תהווה **גרסה עוקבת** לגרסה הקודמת שהוספה , במידה ויש כזו.

פרמטרים:

DS	מצביע למבנה הנתונים.
versionCode	מזהה הגרסה שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL או versionCode<=0
FAILURE	אם versionCode אינו גדול מזה של הגרסה הקודמת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

StatusType AddApplication(void *DS, int appID, int versionCode, int downloadCount)

הוספת אפליקציה חדשה למערכת עם מזהה *appID*, האפליקציה מותאמת לגרסה עם מזהה *versionCode* ויש לה *downloadCount* התקנות.

פרמטרים:	DS	מצביע למבנה הנתונים.
	appID	מזהה האפליקציה שיש להוסיף.
	versionCode	מזהה הגרסה של האפליקציה.
	downloadCount	מספר ההתקנות התחילי של האפליקציה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, appID<=0, versionCode<=0 או downloadCount < 0
	FAILURE	אם קיימת כבר אפליקציה עם מזהה appID או שהגרסה עם מזהה versionCode לא קיימת.
	SUCCESS	במקרה של הצלחה.
סיבוכיות:	$O(log(n)+k)$ במקרה הגרוע, כאשר n הוא מספר האפליקציות ו-k הוא מספר הגרסאות.	

StatusType RemoveApplication(void *DS, int appID)

האפליקציה הוסרה ע"י Gohoo! בעקבות הפרה של תנאי השימוש, יש למחוק אותה מכל מבני הנתונים.

פרמטרים:	DS	מצביע למבנה הנתונים.
	appID	מזהה האפליקציה שצריך להסיר.
ערך החזרה:	INVALID_INPUT	אם DS==NULL או appID<=0
	FAILURE	אם אין אפליקציה עם מזהה appID.
	SUCCESS	במקרה של הצלחה.
סיבוכיות:	$O(log(n)+k)$ במקרה הגרוע, כאשר n הוא מספר האפליקציות במערכת.	

StatusType IncreaseDownloads(void *DS, int appID, int downloadIncrease)

יש לעדכן את מספר ההתקנות של האפליקציה עם מזהה *appID* ולהוסיף לה *downloadIncrease* התקנות.

פרמטרים:	DS	מצביע למבנה הנתונים.
	appID	מזהה האפליקציה שיש לעדכן.
	downloadIncrease	מספר ההתקנות שיש להוסיף לאפליקציה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, appID<=0 או downloadIncrease<=0
	FAILURE	אם אין אפליקציה עם מזהה appID.
	SUCCESS	במקרה של הצלחה.
סיבוכיות:	$O(log(n)+k)$ במקרה הגרוע, כאשר n הוא מספר האפליקציות במערכת.	

StatusType UpgradeApplication(void *DS, int appID)

המפתח שידרג את האפליקציה ויש להעביר אותה **לגרסה העוקבת** של מערכת ההפעלה.

פרמטרים: DS — מצביע למבנה הנתונים.

appID — מזהה האפליקציה שצריך לשדרג.

ערך החזרה: ALLOCATION_ERROR — במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT — אם DS==NULL, appID<=0

FAILURE — אם אין אפליקציה עם מזהה appID, או שלא קיימת גרסה עוקבת.

SUCCESS — במקרה של הצלחה.

סיבוכיות: $O(log(n)+k)$ במקרה הגרוע, כאשר n הוא מספר האפליקציות במערכת.

StatusType GetTopApp(void *DS, int versionCode, int *appID)

יש להחזיר את מזהה האפליקציה עם מספר ההתקנות הגדול ביותר מבין אלו שרצות בגרסה *versionCode*.

אם *versionCode*<0 יש להחזיר את האפליקציה עם הכי הרבה התקנות בכל המערכת

אם אין אפליקציות בגרסה *versionCode* (או במערכת כולה אם *versionCode*<0) יש להחזיר 1- ב-*appID*.

פרמטרים: DS — מצביע למבנה הנתונים.

versionCode — מזהה הגרסה שעבורה נרצה לקבל את המידע.

appID — מצביע למשתנה שיעודכן למזהה האפליקציה המתאימה.

ערך החזרה: ALLOCATION_ERROR — במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT — אם DS==NULL, appID == NULL או versionCode==0.

FAILURE — אם אין גרסה עם מזהה versionCode

SUCCESS — במקרה של הצלחה.

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

אחרת, $O(k)$ במקרה הגרוע. כאשר k הוא מספר הגרסאות.

StatusType GetAllAppsByDownloads(void *DS, int versionCode, int **apps, int *numOfApps)

יש להחזיר את כל האפליקציות בגרסה *versionCode* ממוינות על סמך מספר ההתקנות שלהן. אתם צריכים להקצות את המערך בעצמכם באמצעות **malloc** (כי זה משוחרר בקוד שניתן לכם באמצעות **free**).

אם versionCode<0 מחזיר את האפליקציות בכל המערכת ממוינות על סמך מספר ההתקנות שלהן.

פרמטרים: DS — מצביע למבנה הנתונים.

versionCode — מזהה הגרסה שעבורה נרצה לקבל את המידע.

apps — מצביע למערך שיכיל את כל האפליקציות בגרסה (או במערכת) ממוינות לפי מספר התקנות בסדר **יורד**, אם לשתי אפליקציות יש אותו דירוג אז יש למיין אותן בסדר **עולה** לפי appID.

numOfApps — מצביע למשתנה שיעודכן למספר האפליקציות במערך.

ערך החזרה: ALLOCATION_ERROR — במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT — אם אחד מהפרמטרים שווה ל-NULL או versionCode == 0.

SUCCESS — במקרה של הצלחה.

סיבוכיות: אם versionCode < 0 אז $O(n)$ במקרה הגרוע.

אחרת, $O(n_{versionCode}+k)$ במקרה הגרוע, כאשר $n_{versionCode}$ הוא מספר האפליקציות בגרסה versionCode ו-k הוא מספר הגרסאות.

שימו לב שאתם מקצים את המערך בגודל המתאים!

StatusType UpdateDownloads(void *DS, int groupBase, int multiplyFactor)

החברה גילתה כי עקב טעויות במערכת מתקבלת לעיתים סטייה במספר ההתקנות של אפליקציות מסוימות ויש לבצע עבורן תיקון. לחברת Gohoo! שיטה מיוחדת לקידוד של מידע נוסף בתוך ה-applID. בהינתן *groupBase* האפליקציות שמקיימות $\text{appID} \% \text{groupBase} == 0$ הן אלו ששייכות לקבוצה מסוימת של אפליקציות.

על הפונקציה, בהינתן *groupBase*, לעדכן, עבור כל אפליקציה המתאימה לקבוצה, את מספר ההתקנות שלה ולהכפיל אותו ב-*multiplyFactor*.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	groupBase	מייצג קבוצה של אפליקציות.
	multiplyFactor	הפקטור שבו יש להכפיל את מספר ההתקנות
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, groupBase <1 או multiplyFactor<=0.
	SUCCESS	במקרה של הצלחה.
<u>סיבוכיות</u>:	$O(n+k)$ במקרה הגרוע. כאשר n הוא מספר האפליקציות במערכת ו-k הוא מספר הגרסאות.	

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.

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

<u>סיבוכיות מקום</u>

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

<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.

מבני נתונים 234218 חורף תשע"ה
תרגיל רטוב 1 - מעודכן לתאריך 18/11/2014
עמוד 5 מתוך 6

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 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.
- 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 Statistics, וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C ב library1.h, ממשו את library1.cpp באופן הבא:

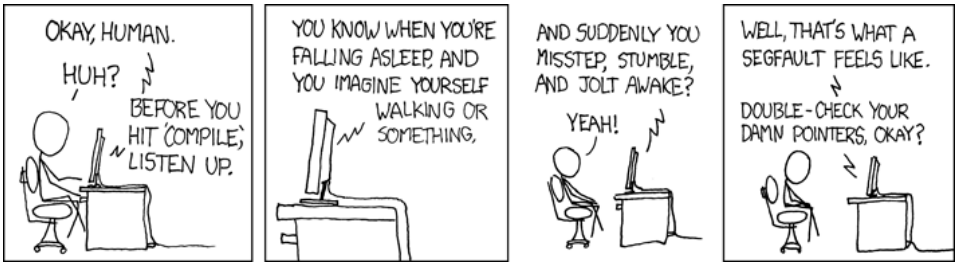
```
#include"library1.h"
#include"Statistics.h"

void* Init() {
        Statistics * DS = new Statistics();
        return (void*)DS;
}
StatusType AddVersion(void *DS, int versionCode){
        return ((Statistics*)DS)-> AddVersion(versionCode);
}
```

וכולי...

- על הקוד להתקמפל על t2 באופן הבא:
  - **g++ –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.



מבני נתונים 234218 חורף תשע"ה
תרגיל רטוב 1 - מעודכן לתאריך 18/11/2014
עמוד 6 מתוך 6

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:

Elad Richardson 012345678 eladrich@cs.technion.ac.il
Majd Omari 123456789 majdo@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!

העתקות בתוכנה תטופלנה בחומרה! (Buh dum tss)

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!