מבני נתונים 234218 חורף תש"ף

גיליון רטוב מספר 1 – מעודכן לתאריך 14.11.2019

תאריך ושעת הגשה: 12/12/2019 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 להפנות למייל: cs234218.technion@gmail.com
- בקשות להגשה מאוחרת יש להפנות באמצעות הטופס האינטרנטי.

הקדמה:

חברת "אוקיינוס דיגיטלי" החליטה לפתח מערכת מחשוב שתעזור בניהול חוות השרתים שבבעלות החברה ברחבי העולם. החברה רוצה מערכת שתאפשר ללקוחות שלה לבקש שימוש בשרתים שונים של החברה בזמן אמת.

בכל חוות שרתים של החברה יש מספר שרתים ועל כל שרת מותקנת מערכת הפעלה כלשהי (Windows או Linux).

המטרה העיקרית של המערכת היא לחכות לבקשה של לקוח של שרת בחווה מסוימת ועם מערכת הפעלה מסוימת והמערכת תקצה ללקוח שרת פנוי. בנוסף, המערכת תאפשר לחברה לעקוב אחר חוות השרתים שהכי משתמשים בהן במערכת הפעלה מסוימת.

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 AddDataCenter(void *DS, int dataCenterID, int numOfServers)

הוספת חוות שרתים חדשה עם מזהה *dataCenterID* שיש בה *numOfServers* שרתים פנויים וממוספרים במספרים עוקבים החל מ-0 (0,1,2,...) וכן הלאה). מערכת ההפעלה של כל השרתים החדשים היא Linux כערך ברירת מחדל.

פרמטרים:

DS	מצביע למבנה הנתונים.
dataCenterID	מזהה חוות השרתים.
numOfServers	מספר השרתים שיש בחווה.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL, numOfServers<=0 או dataCenterID<=0.
FAILURE	אם כבר קיימת חווה עם המספר המזהה הנתון.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(log(n)+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים במערכת ו-m הוא numOfServers.

StatusType RemoveDataCenter(void *DS, int dataCenterID)

מחיקת חוות השרתים עם המזהה *dataCenterID* וכל השרתים שנמצאים בה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL או dataCenterID<=0.
	FAILURE	אם לא קיימת חווה עם המספר המזהה הנתון.
	SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(log(n)+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים במערכת ו-m הוא מספר השרתים בחווה (m הוא 0 אם אין חווה עם המזהה הנתון).

StatusType RequestServer(void *DS, int dataCenterID, int serverID, int os, int *assignedID)

הקצאת השרת *serverID* שבחווה *dataCenterID* והתקנת מערכת ההפעלה *os* עליו.

אם השרת *serverID* תפוס אז המערכת תקצה שרת פנוי אחר שכבר מותקנת עליו מערכת ההפעלה המבוקשת.

אם לא קיים כזה אז המערכת תקצה שרת פנוי עם מערכת הפעלה שונה ותתקין עליו את מערכת ההפעלה המבוקשת.

השרת שהוקצה ייחשב כתפוס ולא יהיה ניתן להקצות אותו שוב עד שישוחרר (באמצעות FreeServer). בנוסף, יש להחזיר את מזהה השרת שהוקצה באמצעות הפוינטר *assignedID*.

יש חשיבות לאופן בחירת השרת שמחזירים במידה והשרת המבוקש תפוס. בהמשך התרגיל יש הסבר לסדר הנדרש ודוגמה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
	serverID	מזהה השרת הנדרש.
	os	סוג מערכת ההפעלה (Linux=0 ו-Windows=1).
	assignedID	מצביע למשתנה אשר מזהה השרת שהוקצה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, serverID<0, serverID>=numOfServers, dataCenterID<=0, os<0 או os>1 או assignedID==NULL. (כאשר numOfServers הוא מספר השרתים בחווה המבוקשת)
	FAILURE	אם לא קיימת חווה עם המזהה המבוקש או שלא נמצא בה אף שרת פנוי.
	SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(log(n))$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים.

StatusType FreeServer(void *DS, int dataCenterID, int serverID)

שחרור השרת *serverID* שבחווה *dataCenterID*. לאחר הפעולה, השרת יהיה פנוי ותישאר עליו מערכת ההפעלה האחרונה שהותקנה עליו.

פרמטרים:	DS	מצביע למבנה הנתונים.
	dataCenterID	מזהה חוות השרתים.
	serverID	מזהה השרת הנדרש.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, serverID<0, serverID>=numOfServers או dataCenterID<=0. (כאשר numOfServers הוא מספר השרתים בחווה המבוקשת)
	FAILURE	אם לא קיימת חווה עם המזהה הנתון או שהשרת עם המזהה הנתון כבר פנוי.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(log(n))$	במקרה הגרוע, כאשר n הוא מספר חוות השרתים.

StatusType GetDataCentersByOS(void *DS, int os, int **dataCenters, int* numOfDataCenters)

החזרת כל המזהים של חוות השרתים ממוינים בסדר יורד על פי מספר השרתים מסוג *os* (לא משנה אם השרת תפוס או פנוי) שיש בחווה. במקרה של שוויון יש למיין מיון משני על פי המזהים בסדר עולה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	os	סוג מערכת ההפעלה (Linux=0 ו-Windows=1).
	dataCenters	מצביע למערך אשר יכיל את מזהי חוות השרתים.
	numOfDataCenters	מצביע למשתנה אשר יכיל את כמות המזהים המוחזרים.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, dataCenters==NULL, numOfDataCenters==NULL, os<0 או os>1.
	FAILURE	אם אין במבנה חוות שרתים בכלל.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(n)$	במקרה הגרוע, כאשר n הוא מספר חוות השרתים.

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

פרמטרים: DS מצביע למבנה הנתונים.

ערך החזרה: אין.

סיבוכיות זמן: $O(n+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים ו-m הוא מספר השרתים הכולל בכל החוות.

סיבוכיות מקום (עבור המבנה וכל הפעולות) - $O(n+m)$ במקרה הגרוע, כאשר n הוא מספר חוות השרתים ו-m הוא מספר השרתים הכולל בכל החוות.

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.

מצביעים הנשלחים לפונקציות על מנת לקבל ערך החזרה לא נקראים כלל בבדיקה האוטומטית אם הוחזר מהפונקציה ערך ששונה מ-SUCCESS ולכן אין חשיבות לערך המוחזר בהם במקרים האלו.

אופן בחירת שרת פנוי כלשהו:

באתחול חוות שרתים חדשה (AddDataCenter) על כל השרתים יהיה מותקן Linux וסדר העדיפות בהחזרת השרתים יהיו בסדר עולה של השרתים (כלומר לשרת הראשון הכי עדיף ואז השני וכן הלאה).

לאחר תפיסת שרת כלשהו (RequestServer) השרת יוצא מסדר העדיפיות כיוון שהוא תפוס.

יש לשים לב שיש צורך בסדר עדיפיות עבור שרתים מסוג Linux וסדר עדיפויות עבור שרתים מסוג WIndows כיוון שלקוח יכול לבקש שרת עם מערכת הפעלה לבחירתו.

לאחר שחרור שרת (FreeServer) השרת ייכנס לסדר העדיפיות (עבור מערכת ההפעלה שהותקנה בו) ויהיה בעל העדיפות הכי נמוכה.

דוגמה:

Init

AddDataCenter 123 5

תיווסף חווה עם 5 שרתים. סדר השרתים יהיה 4<3<2<1<0 (עבור Linux).

RequestServer 123 1 1

זו בקשה לשרת Windows עם עדיפות לשרת מספר 1. כיוון שהשרת פנוי אז יש לבחור את השרת הזה ולהתקין עליו Windows. סדר השרתים יהיה 4<3<2<0 (עבור Linux).

RequestServer 123 1 0

זו בקשה לשרת Linux עם עדיפות לשרת מספר 1. כיוון ששרת 1 תפוס אז יש להחזיר שרת Linux כלשהו. לפי הסדר יש לבחור בשרת מספר 0. סדר השרתים יהיה 4<3<2 (עבור Linux).

RequestServer 123 0 1

זו בקשה לשרת Windows עם עדיפות לשרת מספר 0. כיוון שהשרת תפוס אז יש לבחור שרת Windows כלשהו. כיוון שאין שרתי Windows פנויים יש לבחור שרת Linux פנוי כלשהו ולהתקין עליו Windows. לפי סדר השרתים יש לבחור בשרת 2 ולהתקין עליו Windows. סדר השרתים יהיה 4<3 (עבור Linux).

FreeServer 123 2

שחרור שרת מספר 2. כיוון שמותקן עליו Windows אז הוא שייך לסדר עדיפויות שונה. סדר השרתים יהיה 4<3 (עבור Linux) ו-2 (עבור Windows).

FreeServer 123 1

שחרור שרת מספר 1. בעת שחרור שרת הוא יהיה בעל העדיפות הכי נמוכה. לכן, סדר השרתים יהיה 4<3 (עבור Linux) ו-1<2 (עבור Windows).

FreeServer 123 0

שחרור שרת מספר 0. סדר השרתים יהיה 0<4<3 (עבור Linux) ו-1<2 (עבור Windows).

כעת אם יבקשו שרת Linux, שרת 3 יהיה בעל עדיפות ואם יבקשו שרת Windows, שרת 2 יהיה בעל עדיפות.

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, 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, למשל בשם DataCenterManager , וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C בh.library, ממשו את library.cpp באופן הבא:

```
#include"library.h"
#include"DataCenterManager.h"

void *Init() {
        DataCenterManager *DS = new DataCenterManager ();
        return (void*)DS;
}
StatusType AddDataCenter(void *DS, int dataCenterID, int numOfServers) {
        return ((DataCenterManager  *)DS)-> AddDataCenter(dataCenterID, numOfServers);
}
```

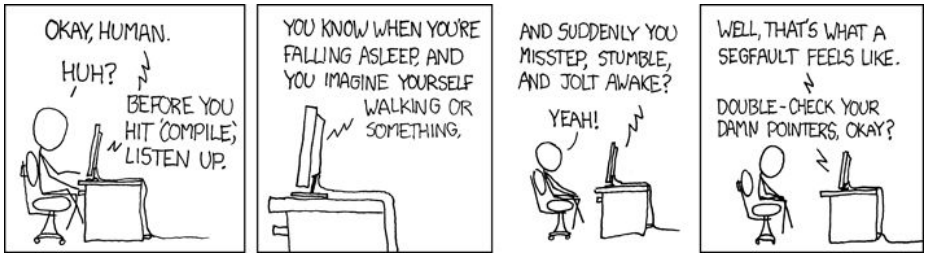
- על הקוד להתקמפל על *csl3* באופן הבא:

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.

מבני נתונים 234218 חורף תש"ף

גיליון רטוב מספר 1 – מעודכן לתאריך 14.11.2019



- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ library.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.**
- יש לתעד את הקוד בצורה נאותה וסבירה.
- 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 roi.bar-zur@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 באמצעות הטופס האינטרנטי. 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!