מבני נתונים 234218 אביב תשע"ט

גיליון רטוב מספר 2 – מעודכן לתאריך 4/6/2019

תאריך ושעת הגשה: 16.06.2019 בשעה 23:59

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

הנחיות:

- 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.
- שאלות על התרגיל יש להפנות למייל: cs234218.technion@gmail.com
- בקשות להגשה מאוחרת יש להפנות באמצעות הטופס האינטרנטי.

הקדמה:

השנה נערכו מספר שינויים בקטלוג הקורסים של הפקולטה. בנוסף להוספה והסרה של קורסים מעניינים חדשים, הוחלט לבצע מיזוג של קורסים מסויימים (לדוגמא – מערכות ספרתיות ותכן לוגי). הפקולטה מעוניינת שהסטודנטים בקורס מבני נתונים יפתחו מערכת לניהול הקורסים, המאפשרת לתחזק את השינויים הללו.

הנכם מתבקשים לממש מבנה נתונים שיאפשר לנהל את מערכת ההרצאות של הפקולטה בכיתות הלימוד השונות בטאוב. לשם פשטות נניח שכל ההרצאות מתקיימות באותו יום בשבוע, וקיימות $m = 10$ שעות שבהן יכולות להתקיים הרצאות. השעות במערכת ממוספרות מ-1 עד m. שימו ❤ – m הוא קבוע לכל אורך התרגיל וכל הפעולות עם סיבוכיות $O(m)$ יחשבו כפעולות עם סיבוכיות $O(1)$! בנוסף, במערכת יש n קורסים, כאשר הפרמטר n ניתן בזמן האתחול של מבנה הנתונים. לכל אורך ריצת התוכנית, ה-ID של הקורסים יהיה מספר בתחום $1, \dots, n$.

הבהרה – בקבוצת הרצאה יכולות להיות מספר הרצאות. לדוגמא, בקורס מסוים יכולה להיות קבוצה 20 שבה יש 3 הרצאות בשעות שונות עם כמות סטודנטים שונה, לדוגמא 20,30,40 סטודנטים. יש 3 הרצאות בקורס ולכן ממוצע הסטודנטים הוא:

$$\frac{20+30+40}{3} = 30\ students\ per\ lecture$$

הפעולות שבהן מבנה הנתונים צריך לתמוך:

void * Init(int n)

מאתחל מבנה נתונים ריק עם n קורסים שה-ID שלהם רץ מ-1 עד n.

פרמטרים: n מספר הקורסים בפקולטה.

ערך החזרה: מצביע למבנה נתונים ריק או $NULL$ במקרה של כישלון.

סיבוכיות זמן: $O(n)$ במקרה הגרוע.

StatusType addRoom(void *DS, int roomID)

הוספת כיתה חדשה עם המזהה *roomID*. בכיתה שנוספה זה עתה לא מתקיימות עדיין הרצאות.

פרמטרים: DS מצביע למבנה הנתונים.

roomID מזהה הכיתה שצריך להוסיף.

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם $DS == NULL$ או $roomID \leq 0$

FAILURE אם roomID קיים.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ בממוצע על הקלט באופן משוערך

StatusType deleteRoom(void *DS, int roomID)

מחיקת הכיתה עם המזהה *roomID*. ניתן למחוק רק כיתה שלא מתקיימות בה הרצאות עדיין.

פרמטרים: DS מצביע למבנה הנתונים.

roomID מזהה הכיתה שצריך למחוק.

ערך החזרה: INVALID_INPUT אם $DS == NULL$ או $roomID \leq 0$

FAILURE אם roomID לא קיים, או שמתקיימות הרצאות בכיתה זו.

	SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ בממוצע על הקלט באופן משוערך

StatusType addLecture(void *DS, int courseID, int groupID, int roomID, int hour, int numStudents)

הוספת הרצאה חדשה. לדוגמא – הרצאה בקורס 234218, בקבוצה 12, שמתקיימת בכיתה (טאוב) 6 ורשומים אליה 60 סטודנטים.

פרמטרים:

DS	מצביע למבנה הנתונים.
courseID	מזהה הקורס
groupID	מזהה הקבוצה
roomID	מזהה החדר
hour	השעה שבה מתקיימת ההרצאה
numStudents	מספר הסטודנטים הרשומים להרצאה

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם $DS == NULL$, $groupID < 0$, $courseID > n$, $courseID < 1$, $numStudents < 0$, $hour \notin \{1, \dots, m\}$, $roomID \leq 0$.
FAILURE	אם קיימת כבר הרצאה של קבוצה זו בשעה הזו, או שהחדר עם המזהה $roomID$ לא קיים או לא פנוי בשעה זו.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* n + \log k)$ בממוצע על הקלט משוערך, כאשר k הוא מספר ההרצאות בקורס ו-n הוא מספר הקורסים במערכת

StatusType deleteLecture(void *DS, int hour, int roomID)

מחיקת ההרצאה שמתקיימת בשעה hour בכיתה roomID

פרמטרים:

DS	מצביע למבנה הנתונים.
hour	השעה שבה מתקיימת ההרצאה
roomID	מזהה החדר שבו מתקיימת ההרצאה

ערך החזרה:

INVALID_INPUT	אם $DS == NULL$, $hour \notin \{1, \dots, m\}$, $roomID \leq 0$
FAILURE	אם לא קיימת כיתה עם מזהה $roomID$ או שלא מתקיימת אף הרצאה בכיתה זו בשעה $hour$.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* n + \log k)$ בממוצע על הקלט משוערך, כאשר k הוא מספר ההרצאות בקורס שאליו שייכת ההרצאה, ו-n הוא מספר הקורסים במערכת.

StatusType mergeCourses(void *DS, int courseID1, int courseID2)

הנהלת הפקולטה החליטה למזג את 2 הקורסים המזוהים. מעתה המזהים courseID1 ו-courseID2 יתייחסו לאותו הקורס וקבוצות ההרצאה של 2 הקורסים יהיו שייכות לאותו קורס. אם בשני הקורסים קיימות קבוצות עם אותו מספר מזהה ולשתיהן יש הרצאות באותה שעה, לא תהיה אפשרות לאחד את הקורסים. דוגמא – לקורס 234247 יש הרצאה של קבוצה 20 בשעה 2, וגם לקורס 236359 יש הרצאה של קבוצה 20 בשעה 2 – לא ניתן לאחד אותם ותוחזר השגיאה המתאימה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	courseID1	מזהה הקורס הראשון
	courseID2	מזהה הקורס השני
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם $DS == NULL$, או $courseID(1/2) \notin \{1, \dots, n\}$
	FAILURE	אם 2 המזהים מתייחסים לאותו הקורס, או שבשני הקורסים יש הרצאות של קבוצות עם אותו מזהה המתקיימות באותה שעה.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(\log^* n + k_1 + k_2)$ משוערך. k_1, k_2 הם כמות ההרצאות בכל אחד מהקורסים המזוהים, ו-n הוא כמות הקורסים במערכת.	

StatusType competition(void *DS, int courseID1, int courseID2, int numGroups, int * winner)

סגל הקורס בכל אחד מהקורסים המזוהים החליט לערוך תחרות ביניהם. כל קורס בוחר את numGroups ההרצאות שרשומים אליהן הכי הרבה סטודנטים. הקורס המנצח הוא הקורס שבו סכום הסטודנטים בכל הרצאות אלה הוא הגדול יותר. אם בקורס אין numGroups הרצאות, כל ההרצאות הקיימות בקורס משתתפות. אם יש תיקו, המנצח הוא הקורס עם ה-ID הגדול יותר. את ה-id של הקורס המנצח יש להחזיר ב-winner.

פרמטרים:	DS	מצביע למבנה הנתונים.
	courseID1	מזהה הקורס הראשון.
	courseID2	מזהה הקורס השני.
	numGroups	מספר ההרצאות מכל קורס שישתתפו בתחרות.
	Winner	מצביע למשתנה שיכיל את ה-ID של הקורס הזוכה.
ערך החזרה:	INVALID_INPUT	אם $DS == NULL$, או $courseID(1/2) \notin \{1, \dots, n\}$ או $numGroups \leq 0$.
	FAILURE	אם שני המזהים מתייחסים לאותו הקורס
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(log^* n + log\ k)$ משוערך, כאשר k הוא מספר ההרצאות המקסימאלי בכל אחד מהקורסים ו-n הוא מספר הקורסים במערכת.	

StatusType getAverageStudentsInCourse(void *DS, int hour, int roomID, float * average)

הפונקציה תחזיר את ממוצע מספר הסטודנטים להרצאה בקורס שיש בו הרצאה בחדר roomID בשעה hour.

פרמטרים:	DS	מצביע למבנה הנתונים.
	hour	השעה שבה מתקיימת ההרצאה
	roomID	מזהה החדר שבו מתקיימת ההרצאה
	average	מצביע למשתנה שיכיל את ממוצע מספר הסטודנטים להרצאה
ערך החזרה:	INVALID_INPUT	אם $DS == NULL$, $hour \notin \{1, \dots, m\}$, $roomID \leq 0$
	FAILURE	אם לא קיימת כיתה עם מזהה roomID או שלא מתקיימת אף הרצאה בכיתה זו בשעה hour.
	SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* n)$ משוערך בממוצע על הקלט, כאשר n הוא מספר הקורסים במערכת.

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: $O(n + k + r)$ במקרה הגרוע, כאשר n הוא מספר הקורסים, k הוא מספר ההרצאות בכל המערכת ו-r הוא מספר החדרים במערכת.

סיבוכיות מקום - $O(n + k + r)$ במקרה הגרוע, כאשר:

- k – מספר ההרצאות במערכת
- n מספר הקורסים במערכת
- r – מספר החדרים במערכת

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.
- 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 `Schedule`, וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C ב library2.h, ממשו את library.cpp באופן הבא:

```
#include "library2.h"
#include "Schedule.h"

void * init(int n) {
      Schedule * DS = new Schedule(n);
      return (void*)DS;
}

StatusType addRoom(void* DS, int roomID) {
      return ((Schedule *)DS)->addRoom(roomID);
{
```

- 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. יש לוודא שהגרסה של g++ היא 4.8.5 ע"י הרצת:

g++ --version

הערות נוספות:

- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ library.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.
- 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 מקרי 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:

Eitan Kosman 012345678 eitan.k@cs.technion.ac.il

Billy Zoom 123456789 billiz@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 שלא תעמוד בקריטריונים הבאים תפסל ותיקנס 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.