מתרגל ממונה על התרגיל: תומר כהן, tomerco20@cs.technion.ac.il

תאריך ושעת הגשה: 13/06/2022 בשעה 23:55

אופן ההגשה: בזוגות. אין להגיש ביחידים. ניתן להיעזר באתר הקורס למציאת שותפים.

הנחיות כלליות:

- שאלות על התרגיל יש לפרסם רק באתר הפיאצה של הקורס תחת לשונית "wet_2"
  - האתר: https://piazza.com/technion.ac.il/spring2022/234218
  - נא לקרוא את השאלות של סטודנטים אחרים לפני שמפרסמים שאלה חדשה, למקרה שנשאלה כבר.
  - שימו לב, תוכן הפיאצה הינו מחייב. כלומר, חובה להתעדכן בעדכונים בתרגיל המפורסמים בפיאצה.
- נא לקרוא את המסמך "נהלי הקורס" באתר הקורס. בנוסף, נא לקרוא בעיון את כל ההנחיות בסוף מסמך זה.
- שימו לב כי הציון לתרגיל זה יכול להיות יותר מ-100. כלומר, הבונוס יכול לתת לתרגיל ציון העולה מעל 100.
- התרגיל מורכב משני חלקים: יבש ורטוב.
  - לאחר קריאת כלל הדרישות, יש לתכנן תחילה את מבני הנתונים על נייר. דבר זה יכול לחסוך לכם זמן רב.
  - לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסיבוכיות בתרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
  - את הפתרון שלכם מומלץ לחלק למחלקות שונות שאפשר לממש (ולבדוק!) בהדרגתיות.
  - המלצות לפתרון התרגיל נמצאות באתר הקורס תחת: "Programming Tips Session".
    - המלצות לתכנות במסמך זה אינן מחייבות, אך מומלץ להיעזר בהן.
  - בתרגיל זה ניתן לממש פונקציית בונוס, פונקציה זו תיבדק רק בטסטים הראשונים, כלומר לא ניתן לתקן אותה במהלך ערעורים על התרגיל.
  - מדיניות הטסטים תבדוק כל פונקציה בנפרד, ובנוסף תבדוק שילוב של מספר פונקציות יחדיו עד כדי כל הפונקציות ביחד. שימו לב, פונקציית הבונוס תיבדק אך ורק בהגשה הראשונית של התרגיל, ובשילוב כלל הפונקציות של התרגיל.
  - שימו לב שציון הבונוס של פונקציית הבונוס, שייך לחלק הרטוב בלבד – כלומר במידה ועברתם את הטסטים של פונקציית הבונוס, בעמודה נפרדת של ציון תקבלו ציון "בינארי" של האם אתם זכאים ל20 נק' נוספות לחלק הרטוב של הגיליון, במידה וכן ציון ה-wet2total יוכל להיות גדול מ-100.
  - מצורף לתרגיל main חלקי לדוגמה, אתם מוזמנים להשתמש בו ולשנות אותו, אך לא להגיש אותו בהגשה, אנחנו נבדוק עם קובץ main שלנו.
- 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.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת: barakgahtan@cs.technion.ac.il.

הקדמה:

כעת לאחר שקבוצת הסטודנטים בוגרי הקורס מבני נתונים התקבלו לעבודה, הם רוצים לאסוף סטטיסטיקות על החברות השונות, ובנוסף, לעקוב אחרי הערך של חברות שנרכשו ושלא נרכשו. בנוסף, לכל עובד חדש בכל חברה יש תקופת חפיפה, בתקופה זו הוא לא מקבל כסף, ולכן המשכורת של כל עובד חדש הינה 0.

כעת אין תנאי על רכישה.

בתרגיל זה, אם חברה c רוכשת חברה אחרת d, אז הערך של חברה c **ושל כל החברות ש-c רכשה בעבר** (בצורה ישירה או עקיפה, כלומר גם חברות שנרכשו ע"י חברות שנרכשו ע"י חברה c, וכן הלאה) עולה בערך של החברה d בנקודת זמן של הרכישה (כלומר, אם חברה d רכשה בעבר חברות, אז ערך זה מתווסף) כפול ערך פקטור אותו מקבלים בקלט בפונקציית הרכישה.

לדוגמא, אם יש 4 חברות:

- חברה 1 עם ערך מקורי 2.
- חברה 2 עם ערך מקורי 1.
- חברה 3 עם ערך מקורי 4.
- חברה 4 עם ערך מקורי 4.

ונניח כי חברה 4 רכשה את חברה 3 עם פקטור $Factor = 0.75$. לאחר מכן, חברה 1 רכשה את חברה 4 עם פקטור $Factor = 0.2$, ולאחר מכן, חברה 1 רכשה את חברה 2 עם פקטור $Factor = 1$.

אזי הערך של כל חברה הינו:

- הערך של חברה 1 הינו 4.4 (הסכום שלה ושל חברה 4 שרכשה כפול 0.2 (כלומר $7 \cdot 0.2 = 1.4$, כי ערך חברה 4 ברכישה היה 7), ביחד עם הערך של חברה 2 שרכשה כפול 1 (כלומר 1)).
- הערך של חברה 2 הינו 1 (הערך המקורי שלה בלבד כי לא רכשה חברות, וחברה 1 לא רכשה חברות לאחר שרכשה אותה).
- הערך של חברה 3 הינו 5 (הסכום של הערך המקורי שלה, ביחד עם הערך המקורי של חברה 2, שנרכשה ע"י חברה אחת לאחר שרכשה את חברה 4 (שרכשה את חברה 3) כפול 1 (כלומר 1)).
- הערך של חברה 4 הינו 8 (הסכום של הערך המקורי שלה, ביחד עם הערך של חברה 3 שנרכשה על ידה כפול 0.75 (כלומר 3), וחברה 2 שנרכשה ע"י חברה 1 לאחר שרכשה את חברה 4 כפול 1 (כלומר 1)).

דרוש מבנה נתונים למימוש הפעולות הבאות:

Void* init(int k)

מאתחל מבנה נתונים עם k חברות הייטק, כל חברה מוגדרת להיות ריקה בהתחלה. הערך המקורי של חברה i הינו i (כלומר, הערך המקורי של חברה 1 הינו 1, הערך המקורי של חברה 2 הינו 2...).

פרמטרים: k מספר חברות ההייטק.

ערך החזרה: מצביע למבנה נתונים ריק או $NULL$ במקרה של כישלון ($k \leq 0$ נחשב ככישלון).

סיבוכיות זמן: $O(k)$ במקרה הגרוע.

StatusType addEmployee(void *DS, int EmployeeID, int CompanyID, int Grade)

הוספת עובד חדש עם מזהה $EmployeeID$ ששייך לחברה $CompanyID$ עם שכר 0 ודרגה $Grade$.

פרמטרים:

DS מצביע למבנה הנתונים.

$EmployeeID$ מזהה העובד שצריך להוסיף.

$CompanyID$ מזהה החברה של העובד.

$Grade$ הדרגה ההתחלתית של העובד.

ערך החזרה:

$ALLOCATION\_ERROR$ במקרה של בעיה בהקצאה/שחרור זיכרון.

$INVALID\_INPU$ אם $DS == NULL$, $EmployeeID \leq 0$, $CompanyID \leq 0$, $CompanyID > k$ או $Grade < 0$.

$FAILURE$ אם קיים כבר עובד עם מזהה EmployeeID.

$SUCCESS$ במקרה של הצלחה.

סיבוכיות זמן: $O(\log^* k)$ משוערך, בממוצע על הקלט, כאשר k הוא מספר החברות במערכת (כפי שהוגדר ב-$init$).

הפונקציה משוערכת ביחד עם הפוקנציות: $acquireCompany$, $companyValue$, $averageBumpGradeBetweenSalaryByGroup$, $sumOfBumpGradeBetweenTopWorkersByGroup$

StatusType removeEmployee(void *DS, int EmployeeID)

העובד בעל המזהה $EmployeeID$ יוצא לפנסיה, וצריך למחוק אותו מהמערכת.

פרמטרים:

DS מצביע למבנה הנתונים.

$EmployeeID$ מזהה העובד שיש למחוק מהמערכת.

ערך החזרה:

$ALLOCATION\_ERROR$ במקרה של בעיה בהקצאה/שחרור זיכרון.

$INVALID\_INPUT$ אם $DS == NULL$ או $EmployeeID \leq 0$.

$FAILURE$ אם אין עובד עם מזהה EmployeeID.

$SUCCESS$ במקרה של הצלחה.

סיבוכיות זמן: $O(\log(n))$ בממוצע על הקלט, כאשר n הוא מספר העובדים במערכת, ו-k הוא מספר החברות במערכת (כפי שהוגדר ב-$init$).

StatusType acquireCompany(void *DS, int AcquirerID, int TargetID, double Factor)

החברה בעלת המזהה $AcquirerID$ רוכשת את החברה $TargetID$.
החברה $TargetID$ מפסיקה להתקיים והעובדים שלה כעת עובדים של $AcquirerID$.
לאחר קריאה לפונקציה זו, שני המזהים $AcquirerID, TargetID$ יתייחסו ל-$AcquirerID$. לדוגמא, אם חברה מספר 4 רוכשת את חברה מספר 6, אז בעתיד כל התייחסות לחברה מספר 4 או 6 תהיה לחברה 4 (לחברה לאחר הקנייה). פרט לפונקציה ***companyValue*** בה ההתייחסות הינה לחברה עצמה.
דוגמא נוספת, אם חברה 3 רוכשת את חברה 4, לאחר מכן חברה 3 רוכשת את חברה 5, ולאחר מכן קיבלנו קלט המקיים $TargetID = 4, AcquirerID = 5$, אזי עבור קלט זה נקבל כי $TargetID == AcquirerID$, מכיוון ששתי החברות מתייחסות לאותה חברה.
שימו לב כי כאן בשונה מהתרגיל הקודם, אנחנו מכפילים בפקטור רק את הערך שאנחנו מוסיפים לחברה $AcquirerID$ ולחברות שאיחדנו עם חברה זו בעבר, ולא מעגלים למטה. וערכים חוקיים ל-$Factor$ הם כל הערכים הגדולים ממש מ-0 (ולא רק גדולים מ-1).

<u>פרמטרים</u>:

DS	מצביע למבנה הנתונים.
$AcquirerID$	מזהה החברה הרוכשת.
$TargetID$	מזהה החברה הנרכשת.
$Factor$	הפקטור שכופלים בו את החברה החדשה.

<u>ערך החזרה</u>:

$ALLOCATION\_ERROR$	במקרה של בעיה בהקצאה/שחרור זיכרון.
$INVALID\_INPUT$	אם $DS == NULL$, $AcquirerID \leq 0$, $AcquirerID > k$, $TargetID \leq 0$, $TargetID > k$, $TargetID == AcquirerID$, $Factor \leq 0.0$
$SUCCESS$	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log^* k + n_{AcquirerID} + n_{TargetID})$ משוערך, כאשר k הוא מספר החברות במערכת (כפי שהוגדר ב-$init$). $n_{AcquirerID}$ ו- $n_{TargetID}$ הם כמות העובדים בחברה הרוכשת והנרכשת, בהתאמה.

הפונקציה משוערכת ביחד עם הפוקנציות: $addEmployee$, $companyValue$, $averageBumpGradeBetweenSalaryByGroup$, $sumOfBumpGradeBetweenTopWorkersByGroup$

StatusType employeeSalaryIncrease(void *DS, int EmployeeID, int SalaryIncrease)

העובד עם מזהה $EmployeeID$ מקבל העלאת שכר בגובה $SalaryIncrease$.

<u>פרמטרים</u>:

DS	מצביע למבנה הנתונים.
$EmployeeID$	מזהה העובד שקיבל העלאה.
$SalaryIncrease$	התוספת לשכר העובד.

<u>ערך החזרה</u>:

$ALLOCATION\_ERROR$	במקרה של בעיה בהקצאה/שחרור זיכרון.
$INVALID\_INPUT$	אם $DS == NULL$, $EmployeeID \leq 0$ או $SalaryIncrease \leq 0$.
$FAILURE$	אם אין עובד עם מזהה $EmployeeID$.
$SUCCESS$	במקרה של הצלחה.

<u>סיבוכיות זמן</u>: $O(\log(n))$ בממוצע על הקלט, כאשר n הוא מספר העובדים במערכת.

StatusType promoteEmployee(void *DS, int EmployeeID, int BumpGrade)

דרגת העובד עם מזהה $EmployeeID$ עולה בערך של $BumpGrade$.

<u>פרמטרים</u>:

DS	מצביע למבנה הנתונים.
$EmployeeID$	מזהה העובד שקיבל העלאה.

$BumpGrade$ העובד עולה דרגה אם פרמטר זה גדול מ-0.

ערך החזרה:

$ALLOCATION\_ERROR$ במקרה של בעיה בהקצאה/שחרור זיכרון.

$INVALID\_INPUT$ אם $DS == NULL$, $EmployeeID \leq 0$.

$FAILURE$ אם אין עובד עם מזהה $EmployeeID$.

$SUCCESS$ במקרה של הצלחה.

סיבוכיות זמן: $O(\log(n))$ בממוצע על הקלט, כאשר n הוא מספר העובדים במערכת.

StatusType sumOfBumpGradeBetweenTopWorkersByGroup (void *DS, int CompanyID, int m, void * sumBumpGrade)

הפעולה מחשבת את סכום הדרגות $(Grades)$ של m העובדים עם המשכורות $(Salary)$ הגבוהות ביותר בחברה עם המזהה $CompanyID$, כאשר אם יש כמה עובדים עם אותה משכורת, נבחר קודם עובדים עם מזהה $EmployeeID$ גדול יותר. אם $CompanyID == 0$ אז הפעולה מחזירה את את סכום הדרגות של m העובדים בעלי המשכורת הגבוהה ביותר מבין **כל העובדים במבנה** (גם כאן, במקרה של שוויון נכריע לפי המזהה של העובדים, קודם עובדים עם מזהה $EmployeeID$ גדול יותר).

לדוגמא, אם יש לנו חברה עם 3 עובדים. לעובד הראשון מתקיים $Grade = 1, Salary = 2, Emplyee\ ID = 4$, לעובד השני מתקיים $Grade = 4, Salary = 10, Employee\ ID = 5$, ולעובד השלישי מתקיים $Grade = 4, Salary = 2, Emplyee\ ID = 6$. וקיבלנו בפונקציה להחזיר עבור החברה את סכום הדרגות של 2 העובדים עם המשכורות הגבוהות ביותר (כלומר, $m = 2$). אזי נחזיר את הערך 8, מכיוון שלעובד השני יש את המשכורת הגבוהה יותר, ולעובדים הראשון והשלישי יש את אותה משכורת, אך לעובד השלישי יש מזהה $(Employee\ ID)$ גדול יותר.

פרמטרים:

DS מצביע למבנה הנתונים.

CompanyID מזהה החברה שעבורה נרצה לקבל את המידע.

m מספר העובדים מהם נרצה לקחת את המידע.

$sumBumpGrade$ מצביע למשתנה שבו תוחזר התוצאה.

ערך החזרה:

$ALLOCATION\_ERROR$ במקרה של בעיה בהקצאת זכרון.

$INVALID\_INPUT$ אם אחד המצביעים שווה ל-NULL או $m \leq 0$, $CompanyID < 0, CompanyID > k$

$FAILURE$ אם מספר העובדים בחברה עם המזהה $CompanyID$ קטן מ-m. ואם $CompanyID = 0$, וגם מספר העובדים בכל המבנה קטן מ-m

$SUCCESS$ במקרה של הצלחה, כלומר כל מצב אחר.

סיבוכיות: $O(\log^*(k) + \log(n))$ משוערך, כאשר k הוא מספר חברות ההייטק ו-n הוא מספר העובדים הכולל במבנה כרגע.

הפונקציה משוערכת ביחד עם הפוקנציות: $addEmployee$, $companyValue$, $acquireCompany$, $averageBumpGradeBetweenSalaryByGroup$

StatusType averageBumpGradeBetweenSalaryByGroup (void *DS, int CompanyID, int lowerSalary, int higherSalary, void * averageBumpGrade)

הפעולה מחשבת את ממוצע הדרגות ($Grades$) של העובדים בעלי משכורת שנמצאת בטווח של [$lowerSalary$, $higherSalary$] (כולל שני הקצוות), בחברה עם המזהה $CompanyID$, אם $CompanyID == 0$ אז הפעולה מחזירה את את ממוצע הדרגות של העובדים בעלי משכורת בטווח [$lowerSalary$, $higherSalary$], מבין **כל העובדים במבנה**.

לדוגמא, אם יש לנו חברה עם 2 עובדים. לעובד הראשון מתקיים $Grade = 1, Salary = 2$, ולעובד השני מתקיים $Grade = 4, Salary = 10$. וקיבלנו בפונקציה להחזיר עבור חברה את ממוצע הדרגות של כל העובדים עם משכורת ($Salary$) בין 0 ל-5 (כולל הקצוות), נחזיר 2.5, ואם היינו מקבלים להחזיר עבור העובדים שהמשכורת שלהם הינה בין 8 ל-12, היינו מחזירים 4.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	CompanyID	מזהה החברה שעבורה נרצה לקבל את המידע.
	$lowerSalary$	המשכורת התחתונה שהחל ממנה אנו סופרים עובדים.
	$higherSalary$	המשכורת העליונה שעד אליה אנו סופרים עובדים.
	$averageBumpGrade$	מצביע למשתנה שבו תוחזר התוצאה.
<u>ערך החזרה</u>:	$ALLOCATION\_ERROR$	במקרה של בעיה בהקצאת זכרון.
	$INVALID\_INPUT$	אם אחד המצביעים שווה ל-NULL או $\text{lowerSalary} < 0, \text{higherSalary} < 0$ $\text{lowerSalary} > \text{higherSalary}$ $\text{CompanyID} < 0, \text{CompanyID} > k$
	$FAILURE$	אם אין עובדים בחברה עם המזהה $CompanyID$ שהרמה שלהם בתחום [lowerSalary, higherSalary]. אם $CompanyID = 0$ אז אם אין עובדים במבנה בתחום [lowerSalary, higherSalary].
	$SUCCESS$	במקרה של הצלחה, כלומר כל מצב אחר.
<u>סיבוכיות</u>:	$O(\log^*(k) + \log(n))$ משוערך, כאשר k הוא מספר חברות ההייטק ו-n הוא מספר העובדים הכולל במבנה כרגע.	

<u>שימו לב</u>, $lowerSalary, higherSalary$ יכולים להיות 0.

הפונקציה משוערכת ביחד עם הפוקנציות: $addEmployee$, $companyValue$, $acquireCompany$, $sumOfBumpGradeBetweenTopWorkersByGroup$

StatusType companyValue(void *DS, int CompanyID, void * standing)

הפעולה מחשבת את הערך של החברה בעלת המזהה $CompanyID$, כפי שהוגדר בתחילת התרגיל. את ערך הפתרון יש לשמור במשתנה $standing$. שימו לב כי כאן בעת המשתנה $CompanyID$ מתכוון לחברה זו, ולא לחברה שרכשה את חברה זו.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	$CompanyID$	מזהה החברה שעבורה נרצה לקבל את המידע.
	$Standing$	מצביע למשתנה שבו תוחזר התוצאה.

ערך החזרה: $ALLOCATION\_ERROR$ במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT אם אחד המצביעים שווה ל-$NULL$ או $CompanyID \leq 0, CompanyID > k$.
SUCCESS במקרה של הצלחה, כלומר כל מצב אחר.

סיבוכיות: $O(\log^*(k))$ משוערך, כאשר k הוא מספר חברות ההייטק.

הפונקציה משוערכת ביחד עם הפוקנציות: $addEmployee$, $acquireCompany$, $averageBumpGradeBetweenSalaryByGroup$, $sumOfBumpGradeBetweenTopWorkersByGroup$

---

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: $O(n+k)$ במקרה הגרוע, כאשר n הוא מספר העובדים ו-k הוא מספר חברות ההייטק.

בונוס (20 נקודות):

StatusType bumpGradeToEmployees(void *DS, int lowerSalary, int higherSalary, int BumpGrade)

הפעולה מעלה את דרגת כל העובדים בכל המבנה שהמשכורת שלהם נמצאת בתחום $[lowerSalary, higherSalary]$ בערך $BumpGrade$.

פרמטרים: DS מצביע למבנה הנתונים.
$lowerSalary$ הערך הנמוך של התחום עבורו נשנה.
$higherSalary$ הערך הגבוה של התחום עבורו נשנה.
$BumpGrade$ הערך בו אנו מגדילים את הדרגה של העובדים בתחום אם פרמטר זה גדול ממש מ-0.

ערך החזרה: $ALLOCATION\_ERROR$ במקרה של בעיה בהקצאת זכרון.
$INVALID\_INPUT$ אם אחד המצביעים שווה ל-$NULL$ או $lowerSalary > higherSalary$, $\text{BumpGrade} \leq 0$
$SUCCESS$ במקרה של הצלחה, כלומר כל מצב אחר.

סיבוכיות: $O(k \cdot \log(n))$ במקרה הגרוע, כאשר k הוא מספר חברות ההייטק ו-n הוא מספר העובדים הכולל במערכת כרגע.

שימו לב, $lowerSalary$ וגם $higherSalary$ יכולים להיות שליליים או לא להכיל עובדים בתחום שלהם, במקרה זה צריך להחזיר $SUCCESS$ (ולשנות רק את העובדים שנמצאים בתחום, אם אין עובדים בתחום, אז לא משנים את הערך לעובדים).

סיבוכיות מקום של כל התרגיל - $O(n+k)$ במקרה הגרוע, כאשר n הוא מספר השחקנים ו-k הוא מספר הקבוצות.

<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.

הנחיות:

חלק יבש:

- **הציון על החלק היבש הוא 15% 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**, ב**++C**, מימוש כזה יאפשר לכם להגיע לפתרון פשוט וקצר יותר לפונקציות אותן עליכם לממש ויאפשר לכם להכליל בקלות את מבני הנתונים שלכם. על מנת לעשות זאת הגדירו מחלקה, נאמר EmployeeManager, וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C ב library2.h, ממשו את library2.cpp באופן הבא:

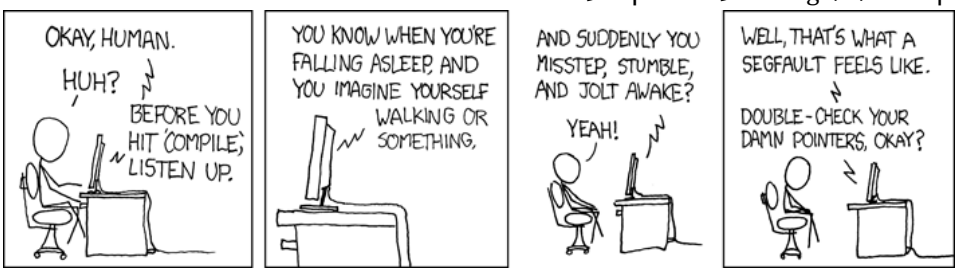
```
#include"library2.h"
#include"EmployeeManager.h"

void* Init() {
        EmployeeManager *DS = new EmployeeManager ();
        return (void*)DS;
}
StatusType AddCompany(void *DS, int CompanyID){
        if (DS == NULL) return INVALID_INPUT;
        return ((EmployeeManager *)DS)-> CompanyID (CompanyID);
}
```

- 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.



הערות נוספות:

- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ library2.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, לא יהיו הגשות חוזרות על הגשות עם include של ספריות שאינן מורשות, אנא בדקו את הקבצים שלכם לפני הגשה.**
- 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:

Tomer Cohen 012345678 tomerco20@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 למתרגל האחראי בלבד בכתובת barakgahtan@cs.technion.ac.il.
  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!