מבנה נתונים

תרגיל בית מס' 3 – תכנות

אנא הקפידו להגיש לפי הנחיות ההגשה שבאתר

מועד ההגשה: 23.02.2024

בתרגיל בית זה, תתבקשו לענות לשאלה בצורה תאורטית. כלומר, יש לתאר מהו הפתרון שאתם מציעים מילולית. יש להוכיח את הסיבוכיות של הפתרון שלכם. לאחר מכן, יש לתכנת את הפתרון שאתם מציעים, וכמובן לעמוד בדרישות הסיבוכיות. אי-עמידה בדרישת הסיבוכיות תגרור הורדת ניקוד.
חלוקת הניקוד:

- תיאור מילולי נכון ומשכנע והוכחת סיבוכיות שווה 30% מהציון.
- הטסטים והבדיקה הידנית של הקוד שווים 70% מהציון.
- כתיבת קוד שלא תואם לסיבוכיות ישפיע שלילית על הציון.

הערות חשובות:

- אתם תתכנתו בשפת C. אין להשתמש בשפות תכנות אחרות.
- תמיד נכניס קלטים חוקיים לקריאות לפונקציה.
- הקובץ אשר תגישו לא יכלול $main()$ בכלל (ואם יכלול אז נמחק אותה). אנחנו נשים $main$ משלנו ונבדוק את הקוד שלכם על זה.
- אין לשנות את חתימות הפונקציות שניתנו לכם **בכלל**, שינוי כזה עלול להכשיל את כל הבדיקות האוטומטיות.
- מותר לכם להגדיר מספר בלתי מוגבל של פונקציות עזר.
- מותר להשתמש בספריות הבאות **בלבד**: stdlib.h,stdio.h .
- נא לשים את תעודות הזהות של שני השותפים בתחילת קובץ C כהערה ב-C.
- אתם רשאים להחליט איך לפתור את השאלה בכל צורה שנוחה לכם המשתמשת בכל מה שנלמד בהרצאות ובתרגולים עד כה. אתם יכולים להחליט להשתמש בכל מבני הנתונים הנלמדו או בשילובים שלהם או בווריאציות שונות שלהם כל עוד עדיין תעמדו בדרישות השאלה. (החומר הנלמד הינו עד **עצי חיפוש**, כולל)
- מאוד חשוב לפשט את הקוד ככל הניתן. הגדירו פונקציות איפה שהגיוני; זה יעזור לכם לתכנת בצורה תקינה וחסרת באגים ומקל עליכם מאוד את דיבוג החלקים הלא עובדים בקוד. **אסור לשכפל הקוד!**
- יש לתעד את הקוד היטב: לכל פונקציה שאתם מגדירים הגדירו מהו הקלט, הפלט והסיבוכיות שלה. בנוסף לזה, הסבירו בקצרה איך היא עובדת אם זה לא ברור מהסתכלות מהירה על הפונקציה. <u>הערה כזו יש להוסיף אותה **מעל וצמוד** להגדרה של הפונקציה.</u>
- השתמשו בשמות משמעותיים לפונקציות ולמשתנים שלכם לאורך כל התרגיל.
- **בונוס:** בכדי לעודד כתיבת קוד יפה, פשוט, ברור וקריא, יתווספו עד 10+ נקודות לחלק התכנותי לשיקול הבודק.
- **חשוב:** ודאו שבהגשה במודל הקוד שלכם אינו מבצע הדפסות, ואינו מבקש קלטים מהמשתמש. מספיק להגיש את הממשק של הקוד. כל ההדפסות והקליטות יקרו ב-$main$ שאותה אתם לא תממשו.
- **חשוב:** יש לשחרר את הזיכרון שאתם כבר לא תשתמשו בו.
- **אי התייחסות להערות אלה עלולה להוריד ניקוד ולהאריך את זמן בדיקת התרגיל.**

- **העתקות יטופלו בחומרה בוועדת משמעת אוניברסיטאית.** בנוסף לבדיקה הידנית, תבוצע גם בדיקה אוטומטית בין הפתרונות של עמיתיכם ובין הפתרונות המוגשים מול מקורות מהאינטרנט.

<u>כיצד נממש מבנה הנתונים בשפת C?</u>

מותר לכם להגדיר את מבנה הנתונים כרצונכם, בבדיקה אנחנו רק מניחים שמותר לנו להשתמש בממשק שנתון לכם (ההצהרות של הפונקציות מופיעות בהמשך, אותם אסור לשנות בכדי שהטסטים יעברו).
בתחילת הקובץ שנתון לכם תראו את המקטע הבא:

```
typedef struct DataStructure {

} DataStructure;
```

מותר לכם להוסיף איזו שדות ומבני הנתונים שתרצו. זהו מבנה הנתונים אשר נשתמש בו לאורך כל התרגיל. הטסטים לא מניחים כלום על שמות השדות ואיזה מבנים השתמשתם, ככה שאם המימוש נותן פלט נכון, ככה גם הטסטים יהיו נכונים. מימוש התרגיל גם ייבדק ידנית כדי לוודא שהמימוש שלכם עומד בזמני הריצה הדרושים. שימו לב, המבנה הזה יוכל להחזיק כל מה שתרצו.

לדוגמא, נניח שמבנה הנתונים שתציעו אותו מחזיק שני מערכים דינאמיים ושדה מסוג int המציין מספר הגישות למבנה הנתונים. ניתן לעשות זאת ע"י:

```
typedef struct DynamicArray {
    int currentSize;
    int emptyIndex;
    int* array;
} DynamicArray;

typedef struct DataStructure {
    int accessAmount;
    DynamicArray firstArray;
    DynamicArray secondArray;
} DataStructure;
```

כעת, אם תרצו להעביר משתנה מטיפוס $DataStructure$, אז בעצם אתם מעבירים את כל המבנה: שני המערכים ואת השדה שהוספתם. אפשר להכליל את ההסבר הזה לכל צורך שלכם.

כמובן, בדוגמא הזו, עדיין צריך לממש את הפונקציות שמערך דינאמי תומך בהם בכדי להשתמש בממשק של מערך דינאמי, ויש לממש את הממשק של $DataStructure$ שבעצם הולך להיות מבנה הנתונים שתרצו להשתמש בו בכדי לפתור את השאלה.

<u>השאלה:</u>

באתר מסוים למכירת מוצרים חשמליים, החליטו לממש <u>מערכת מעקב</u> אחרי איכות הייצור של הפריטים. כל פריט מוגדר על ידי מדד האיכות שלו $quality$, והזמן שנכנס <u>למערכת המכירות באתר</u> $time$. כל הנתונים האלה הינם מספרים שלמים.

- בכל נקודת זמן $time$, מכניסים לכל היותר מוצר אחד. (כלומר לא יכולים להיות שני מוצרים בתוך מבנה הנתונים שיש להם את אותו $time$).
- כל מוצר מוכנס לכל היותר פעם אחת למנה הנתונים.
- המוצרים לא בהכרח מוכנסים <u>למערכת המעקב</u> בסדר עולה, לא לפי איכות ולא לפי זמן הכניסה לאתר.
- **<u>הגדרה:</u>** יהיו $x_1, \dots, x_n$ המוצרים שנרשמו כרגע במערכת, שנכנסו בזמנים $t_1 < t_2 < \cdots < t_n$ בהתאמה. ויהי $x_{i_1} \le x_{i_2} \le \cdots \le x_{i_n}$ מיון המוצרים לפי האיכות שלהם $quality$, אם יש שני מוצרים שיש להם את אותו ערך של איכות אז נמיין אותם בין עצמם לפי זמני הכניסה שלהם. נגדיר המוצר ה-"j-איכותי" להיות x_{i_j}, כלומר המוצר שיקבל את האינדקס $j-1$ במערך הממוין (בגלל שטכנית מערכים ב-C מתחילים מ-0).

<u>שימו לב:</u> אם לשני מוצרים יש אותה איכות אז נמיין בין איכויות אלה לפי זמן הכניסה של המוצר. ניזכר שמובטח לנו שזמני הכניסה שלהם שונים. **מסקנה: יש מיון אחד ויחיד של המוצרים תחת הנחות אלה.**

<u>לדוגמא:</u> אם נכניס 5 מוצרים x_1, x_2, x_3, x_4, x_5 (משמאל לימין) עם הזמנים 1,2,3,4,5 בהתאמה והאיכותיות $2,4,-1,18,2$ בהתאמה אז המיון <u>היחידי</u> שלהם יניב את המערך הממוין $x_3 \le x_1 \le x_5 \le x_2 \le x_4$. ולכן האיבר ה-"1-איכותי" הוא x_3 והוא המוצר הכי גרוע שנמצא כרגע במבנה, המוצר ה-"2-איכותי" יהיה x_1, המוצר ה-"3-איכותי" יהיה x_5, המוצר ה-"4-איכותי" יהיה x_2, והמוצר ה-"5-איכותי" יהיה x_4 והוא המוצר הכי טוב שנמצא כרגע. נשים לב שהוחלט שמתקיים $x_1 \le x_5$ למרות ששניהם בעלי אותה איכות, הסיבה לכך היא שמיינו אותם לפי זמני הכניסה שלהם (ל-x_1 יש זמן כניסה 1 שקטן מזמן הכניסה שיש ל-x_2 השווה 2).

<u>שימו לב:</u> יכולנו להכניס את אותם מוצרים בדיוק עם אותם זמנים ועם אותם איכותיות אבל בסדר שונה: דבר זה לא היה משנה את המיון היחידי שלהם שיכול להתקבל. למשל יכולנו להכניס אותם בסדר הזה x_2, x_4, x_1, x_5, x_3 (משמאל לימין) למערכת המעקב (כאשר עדיין x_1 יוכנס למערכת עם המידע שזמן הכניסה שלו 1 והאיכות שלו 2, ועדיין x_2 יוכנס למערכת עם המידע שזמן הכניסה שלו 2 והאיכות שלו 4, וכו...).

<u>שימו לב</u>: בדוגמא בהמשך המוצר ה-"j-איכותי" תורגם לאנגלית כ-
"$j-best\ product$"

לצורך הפשטות, אנחנו מניחים שזמנים ואיכות שניהם אך ורק מטיפוס int, ואפילו **חיוביים**.
הנחות אלו נועדו כדי לפשט את המימוש קצת. הממשק מופיע מטה, אין לשנות אותו בכלל.
בדרישת סיבוכיות זמן מסמנים ב-n את מספר האיברים הנמצאים במבנה הנתונים.

הממשק הינו:

1) $DataStructure\ Init(int\ s)$: אתחל מבנה נתונים ריק, עם מספר מיוחד s המתאר איכות מאוד מבוקשת של מוצרים. מחזירים את מבנה הנתונים אשר אותחל.
<u>סיבוכיות זמן:</u> $O(1)$ במקרה הגרוע.
2) $void\ AddProduct(DataStructure *\ ds, int\ time, int\ quality)$: הכנס מוצר חדש למבנה הנתונים ds (כלומר למערכת המעקב). המוצר בעל איכות $quality$ וזמן הכנסה $time$. ניתן להניח שלא מכניסים שני מוצרים שונים באותו זמן $time$.
<u>סיבוכיות זמן:</u> $O(\log(n))$ במקרה הגרוע.
3) $void\ RemoveProduct(DataStructure *\ ds, int\ time)$: הסר את המוצר אשר הוכנס בזמן $time$ ממבנה הנתונים ds. (אם אין מוצר כזה לא צריך לבצע דבר)
<u>סיבוכיות זמן:</u> $O(\log(n))$ במקרה הגרוע.
4) $void\ RemoveQuality(DataStructure * ds, int\ quality)$: הסר את **כל** המוצרים אשר בעל איכות $quality$ ממבנה התנותים ds. (אם אין מוצרים כאלה לא צריך לבצע דבר).
<u>סימון:</u> נסמן ב-k את מספר האיברים בעלי האיכות $quality$ במבנה הנתונים
<u>סיבוכיות זמן בפתרון התאורטי:</u> $O(\min(\mathrm{klog(n)}, \mathrm{n})$.
<u>סיבוכיות זמן בפתרון התכנותי:</u> $O(k\mathrm{log(n)})$.
5) $int\ GetIthRankProduct(DataStructure\ ds, int\ i)$: החזר את <u>זמן ההכנסה</u> של המוצר ה-"i-איכותי", אם אין כזה החזר -1.
<u>סיבוכיות זמן:</u> $O(\log(n))$ במקרה הגרוע.
6) $int\ GetIthRankBetween(DataStructure\ ds, int\ time1, int\ time2, int\ i)$: הסתכל על כל הפריטים שהוכנסו בין t_1 (כולל) עד t_2 (כולל), והחזר את <u>זמן ההכנסה</u> של הפריט ה-"i-איכותי" מביניהם, אם אין כזה החזר -1.
<u>סיבוכיות זמן בפתרון התאורטי:</u> $O(i\mathrm{log}(n))$ במקרה הגרוע.
<u>סיבוכיות זמן בפתרון התכנותי:</u> ניתן להניח במימוש שתמיד $i \leq 10$ תמיד, ולכן הדרישה במקרה זה תהיה $O(\mathrm{log(n)})$ במקרה הגרוע. חובה להסביר בהגשה את הפתרון שממומש בחלק התאורטי וגם בחלק התכנותי.
7) $int\ Exists(DataStructure\ ds)$: מחזיר 1 אם יש מוצר בעל איכות s, אחרת מחזיר 0.
<u>סיבוכיות זמן:</u> $O(1)$ במקרה הגרוע.

<u>סיבוכיות מקום של כל המבנה:</u> אם יש n מוצרים, אז $O(n)$ מקום נתפס ע"י כל המבנה.
<u>שימו לב:</u> יתכן בטסטים שנשתמש בזמנים או איכותיות אשר לא נמצאים במבנה, זהו קלט חוקי. נשתמש רק במספרים חיוביים בטסטים.

דוגמא:

DataStructure ds = Init(11) // initializes an empty data structure

AddProduct(&ds, 4, 11) // Adds a product at time t=4 and quality q=11

AddProduct(&ds, 6, 12) // Adds a product at time t=6 and quality q=12

AddProduct(&ds, 2, 13) // Adds a product at time t=2 and quality q=13

AddProduct(&ds, 1, 14) // Adds a product at time t=1 and quality q=14

AddProduct(&ds, 3, 15) // Adds a product at time t=3 and quality q=15

AddProduct(&ds, 5, 17) // Adds a product at time t=5 and quality q=17

AddProduct(&ds, 7, 17) // Adds a product at time t=7 and quality q=17

GetIthRankProduct(ds, 1) //The i=1 best product has time t=4 and quality q=11,returns 4

GetIthRankProduct(ds, 2) //The i=2 best product has time t=6 and quality q=12,returns 6

GetIthRankProduct(ds, 6) //The i="6 best product" has time t=5 and quality q=17,returns 5

GetIthRankProduct(ds, 7) //The i="7 best product" has time t=7 and quality q=17,returns 7

GetIthRankBetween(ds, 2, 6, 3) // looks at values with time {2,3,4,5,6} and returns the i="3 best product" between them, which has time t=2.

Exists(ds) // returns 1, since there exists a product with quality q=s=11

RemoveProduct(&ds, 4) // removes product with time t=4 from the data structure

Exists(ds) // returns 0, since there is no product with quality q=s=11

בהצלחה!!