סדנת תכנות בשפת C ו-++C (67312)

תרגיל 2

תאריך ההגשה של התרגיל: יום א' 20.11.22 - עד השעה 23:59

נושאי התרגיל: struct, פוינטרים, ניהול זיכרון וקריאה מקובץ

הקדמה

האוניברסיטה העברית יצאה בהחלטה חדשה: כל סטודנט יקבל מחשב נייד במתנה. לצורך כך הוקם מחסן ובו מוחזקים המחשבים המיועדים לחלוקה. אתם אחראיים לנהל את המחסן הזה. אתם מעוניינים להחזיק מבנה נתונים השומר את שמות הדגמים הנמצאים במחסן ואת הכמות המדויקת הקיימת מכל דגם. כמו כן אתם מעוניינים לבצע בקלות יחסית פעולות כמו הוספה של דגם חדש, חיפוש ועדכון כמות של דגם קיים. לצורך כך נשתמש **בעץ חיפוש בינארי**.

עץ חיפוש בינארי

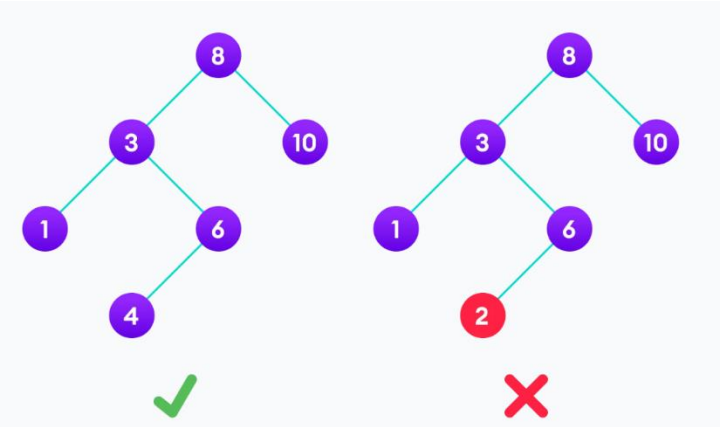
עץ בינארי הוא מבנה נתונים מסוג עץ שבו לכל קודקוד יש 2 בנים לכל היותר.

עץ חיפוש בינארי הוא עץ בינארי שבו לכל קודקוד מתקיים הכלל הבא:

עבור קודקוד כלשהו V –

- ✓ כל הקודקודים שנמצאים בתת העץ הימני של V גדולים ממנו.
- ✓ כל הקודקודים שנמצאים בתת העץ השמאלי של V קטנים ממנו.

הערה: המושגים "קודקוד גדול" או "קודקוד קטן" מצריכים הגדרת יחס סדר על המידע השמור בתוך הקודקודים. בהמשך נציין באיזה יחס סדר נשתמש בתרגיל זה, על מנת למיין את המידע השמור בקודקודי העץ.



כפי שניתן לראות בתמונה[1]:
העץ השמאלי הינו עץ חיפוש בינארי, ואילו העץ הימני, אינו עץ חיפוש בינארי משום שבתת העץ הימני של קודקוד 3, יש קודקוד שקטן ממנו (2).

[1] התמונה נלקחה מהאתר https://www.programiz.com/dsa/binary-search-tree

כעת נתאר מספר פעולות, הרלוונטיות עבורנו, שניתן לבצע על עץ חיפוש בינארי.

חיפוש

מציאת קודקוד בעץ המכיל ערך נתון.

אלגוריתם:

1. נעדכן את השורש להיות הקודקוד הנוכחי.
2. נשווה את הערך של הקודקוד הנוכחי לערך אותו אנו מחפשים:
   א. אם הערך זהה- נחזיר את הקודקוד הנוכחי.
   ב. אם הערך המבוקש גדול מערכו של הקודקוד הנוכחי- נקבע את הבן הימני להיות הקודקוד הנוכחי ונחזור לשלב 2.
   ג. אם הערך המבוקש קטן מערכו של הקודקוד הנוכחי- נקבע את הבן השמאלי להיות הקודקוד הנוכחי ונחזור לשלב 2.

הוספה

הוספת ערך מסוים לעץ תוך שמירה על מבנה עץ החיפוש.

אינטואיציה:

בצורה דומה לתהליך החיפוש, נרד במורד העץ ימינה או שמאלה בהתאם לערכו של הקודקוד אותו אנו מעוניינים להוסיף. כאשר לא ניתן להמשיך לרדת מובטח שזהו מיקומו המתאים של הקודקוד החדש.

אלגוריתם:

1. נעדכן את השורש להיות הקודקוד הנוכחי.
2. אם העץ ריק (אין בו קודקודים)- ניצור קודקוד עם הערך הרצוי.
3. נשווה את הערך של הקודקוד הנוכחי לערך אותו אנו מוסיפים:
   א. אם הערך זהה- לא נעשה דבר (הערך כבר קיים בעץ)
   ב. אם הערך החדש גדול מערכו של הקודקוד הנוכחי- נקבע את הבן הימני להיות הקודקוד הנוכחי ונחזור לשלב 2.
   ג. אם הערך החדש קטן מערכו של הקודקוד הנוכחי- נקבע את הבן השמאלי להיות הקודקוד הנוכחי ונחזור לשלב 2.

שימו לב: ההוספה אינה משנה קודקודים אחרים הקיימים כבר בעץ. כלומר, קודקוד חדש תמיד יתווסף בתור בן לקודקוד שלפחות אחד מבניו לא קיים.

מחיקה

הסרת קודקוד בעל ערך מסוים מתוך העץ תוך שמירה על מבנה עץ החיפוש.

אלגוריתם:

נחלק למקרים הבאים –

1. אם לקודקוד אין ילדים בכלל – נוכל פשוט להסיר קודקוד זה.
2. אם לקודקוד יש ילד אחד בלבד - נעדכן ילד זה להיות הבן של אביו של הקודקוד אותו אנו מעוניינים למחוק (תחשבו על זה כך: הסבא מאמץ את הנכד להיות בנו).

3. אם לקודקוד יש שני ילדים - עלינו למצוא את האיבר העוקב של קודקוד זה.

עוקב (successor) של קודקוד Z בעץ הוא הקודקוד בעל הערך הקטן ביותר הגדול מZ. כלומר, האיבר המינימלי באוסף האיברים הגדולים מ Z.

איך נמצא את העוקב? נחפש את האיבר המינימלי בתת העץ הימני של Z (כאמור, אוסף האיברים הגדולים מ Z נמצאים בתת העץ הימני).

מציאת איבר מינימלי בתת עץ תתבצע על ידי ירידה שמאלה עד הסוף (עד שלא ניתן לרדת שמאלה).

לאחר שמצאנו את העוקב, נשים את ערכו בקודקוד אותו אנו מעוניינים למחוק, ונמחק את העוקב. מובטח לנו שלעוקב אין בן שמאלי (משום שהוא האיבר המינימלי בתת העץ..) ולכן המחיקה של העוקב תתבצע בפשטות (כלומר חזרנו לשלב 1 או 2).

לקריאה נוספת על עץ חיפוש בינארי: https://www.programiz.com/dsa/binary-search-tree

לוויזואליזציה של עץ חיפוש בינארי:

https://www.cs.usfca.edu/~galles/visualization/BST.html

בניית העץ

לאחר שהבנו בצורה כללית מהו עץ חיפוש בינארי, נפרט כעת כיצד הוא ממומש בתרגיל זה.

בתרגיל זה נשתמש במבנה בשם: struct Product המייצג מוצר כלשהו. במקרה שלנו, מבנה זה מייצג מחשב נייד, ומחזיק את שם הדגם והכמות הקיימת ממנו.

קודקוד בעץ מיוצג על ידי struct Node, המכיל שני שדות מסוג Node* עבור הבן הימני והבן השמאלי, ושדה מסוג struct Product עבור המוצר.

יחס הסדר המוגדר על קודקודי העץ הוא סדר אלפביתי של שמות המוצרים. לדוג'

Dell Xps > Lenovo wikiwiki > Mac > Macbook > Macbook pro banana

הערה: סדר לקסיקוגרפי ממומש באמצעות הפונקציה strcmp.

קובץ h

עליכם לעקוב אחר קובץ הheader בשם ProductBst.h המצורף ולממש את הפונקציות המופיעות בו. המימוש יהיה בקובץ C בעל אותו שם (עם סיומת שונה).

קובץ h הינו קובץ המכיל הצהרות על פונקציות, structs וקבועים גלובליים. קובץ זה מצהיר אך ורק על פונקציות ומשתנים שאנחנו רוצים שיהיו משותפים למספר קבצים. כלומר, רק פונקציות שאנחנו רוצים שיהיו "זמינות לכולם". קובץ h מגדיר לנו API – Application Programming Interface.

לכן, **אסור להוסיף** לקובץ ה-h פונקציות שמימשתם בקובץ ProductBst.c לטובת פתרון התרגיל שאינן נכללות בקובץ ה-h שקיבלתם. מאחר ואינן חלק מממשק התוכנית שלכם ואינכם רוצים שיהיו זמינות לשימוש מחוץ לקובץ ProductBst.c . באופן כללי, נא לא לשנות את קובץ ה-h. שינוי קובץ זה עלול לגרור כישלון בתרגיל.

הנחות על הקלט

בין שאר הפונקציות, עליכם לממש את הפונקציה build_bst המקבלת כקלט שם של קובץ, קוראת ממנו את שמות המוצרים והכמויות ובונה את העץ.

אתם יכולים להניח שקובץ הקלט יהיה תקין ותואם לפורמט הבא בדיוק:

- ✓ בקובץ תהיה **לפחות** שורה אחת ולא יהיו בו שורות ריקות.
- ✓ כל שורה מייצגת מוצר.
- ✓ כל שורה תהיה מהפורמט הבא: <product_name>: <quantity>

כאשר product_name הוא שם המוצר (כל מחרוזת היא חוקית) ו- quantity היא הכמות הקיימת במלאי מאותו המוצר. לדוג':

Dell Xps Zt15: 20

Lenovo Wana8Banna: 10

Mac ProPeanuts: 1

*שימו לב! בין שם המוצר לבין הנקודותיים **לא** קיים רווח.*

- ✓ ניתן להניח כי הכמות הינה מספר שלם (בהמשך נסביר איך לטפל במספרים שליליים ואפס).

- ✓ אין להניח כי בכל שורה, לאחר שם המוצר והכמות שלו, לא מופיעים תווים נוספים בשורה זו. אם יש שורה כזו, כלומר שורה מהצורה:

  <product_name>: <quantity> blallla..

  יש להדפיס הודעת שגיאה ולא להוסיף דבר.
- ✓ אין להניח כי כל מוצר מופיע פעם אחת בדיוק. במקרה בו מנסים **להוסיף** לעץ מוצר ששמו כבר קיים בעץ, התוכנית תדפיס הודעת שגיאה ולא תוסיף דבר.

  (באחריות הפונקציה add_product)
- ✓ אין להניח כי הכמות היא מספר **חיובי**. במקרה שמנסים **להוסיף** לעץ מוצר עם כמות שלילית או אפס, יש להדפיס הודעת שגיאה ולא להוסיף דבר. (באחריות הפונקציה add_product).

דגשים והנחיות לתרגיל

- ✓ שימו לב! אין פונקציית main בפתרון שלכם. הגשה של פונקציית main תגרור כישלון ב-presubmit וכתוצאה מכך ציון 0 בתרגיל.

- ✓ תחת התרגיל במודל קיים קובץ בשם main.c שיעזור לכם לבדוק את הפונקציות שאתם כותבים. שימו לב, זהו קובץ בסיסי ביותר ואנו מציעים בחום לבנות טסטים מורכבים יותר. כדי להפעיל אותו עם Clion עליכם לשים אותו באותה תיקיה של התרגיל שלכם ולוודא שהוא מופיע ברשימת הקבצים של CMakeLists.txt.

- ✓ בתרגיל זה אסור להשתמש בפונקציה exit כלל!

בכל פונקציה אותה אתם מממשים, עליכם לוודא שבמידה ויש כישלון כלשהו בתוכנית, למשל הקצאת זיכרון שנכשלה, מודפסת הודעת שגיאה, ולאחר מכן משוחרר הזיכרון והתוכנית מסתיימת.

כל פונקציה משחררת את הזיכרון שהיא הקצתה בלבד! כלומר, אם פונקציה מקבלת כפרמטר מצביע ל- Node ומתרחשת שגיאה במהלך הרצת הפונקציה, אין באחריותה של הפונקציה לשחרר את הזיכרון של *Node שהיא קיבלה, שכן, היא לא זו שהקצתה את זיכרון זה ולכן היא אינה מנהלת אותו.

- ✓ זהו התרגיל הראשון בו אתם מנהלים זיכרון. שימו לב שאין לכם דליפות זיכרון!

בתרגיל זה נוריד מספר לא מבוטל של נקודות על דליפות זיכרון - ולא יתקבלו ערעורים על מספר הנקודות שיורדו.

למשל, אם פונקציית שחרור המשאבים שלכם (delete_tree) גורמת לדליפת זיכרון, ירדו נקודות בכל טסט.

- ✓ בכדי לוודא שאין דליפות זיכרון בתרגיל, עליכם להשתמש ב- valgrind, תוכנה הבודקת את ניהול הזיכרון שלכם. הוראות לשימוש בכלי זה תוכלו למצוא במודל הקורס. אנו מריצים את valgrind עם הדגל --leak-check=full (שימו לב: יש שני מקפים לפני ה leak).

כל הערה שעולה מהרצה של valgrind תיחשב כשגיאה בניהול הזיכרון!

דוגמה להרצה:

```
valgrind ex2 --leak-check=full
```

- ✓ על הודעות השגיאה להיות מודפסות ל stderr. מבנה הודעת השגיאה הוא תחילית " ERROR:" ולאחר מכן יופיע מסר המפרט את הבעיה. **הנוסח המדויק של כל סוג הודעה נמצא בקובץ ה h.** עליכם לבחור את הודעת השגיאה המתאימה עבור כל שגיאה. **אין לשנות מנוסח זה.**

- ✓ הימנעו משימוש במשתנים גלובלים.

- ✓ אין להשתמש בפונקציה atoi.

פתרון בית ספר

- ✓ פתרון בית הספר, מקומפל עם הקובץ main.c המסופק לכם במודל, זמין בנתיב:

```
~labcc/www/ex2/schoolSolutionWithMain
```

כדי להריץ את הפתרון עם קובץ קלט, עליכם להריץ את הפקודה

```
~labcc/www/ex2/schoolSolutionWithMain <path_to_input_file>
```

נהלי הגשה

- ✓ קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים שבאתר הקורס.
  כמו כן, זכרו כי התרגילים מוגשים ביחידים וגם העבודה עליהם צריכה להיות ביחידים!
  אנו רואים העתקות בחומרה רבה!

- ✓ הגשת התרגיל תתבצע באמצעות ה- git והקבצים היחידים שיש להגיש הם: ProductBst.c, ProductBst.h. הגשה של כל קובץ אחר תוביל לכישלון בתרגיל.

- ✓ בשפת C יש פונקציות רבות המיועדות לעבודה עם קלט. אין צורך להמציא מחדש את הגלגל!
  לפני תחילת העבודה על התרגיל, מומלץ לחפש באינטרנט את הפונקציות המתאימות ביותר לקבלת קלט מהמשתמש, להדפסת קלט, עיבוד קלט מסוגים שונים וכו'.
  ודאו שכל הפונקציות שבהן אתם משתמשים מתאימות ל standard C99, וכי אתם יודעים כיצד הן מתנהגות בכל סיטואציה.
- ✓ אנא וודאו כי התרגיל שלכם עובר את ה- Pre-Submission Script ללא שגיאות או אזהרות. קובץ ה- Pre-submission Script זמין בנתיב:

```
~labcc/presubmit/ex2/run
```

בכדי להריץ אותו על התרגיל שלכם, עליכם לייצר קובץ tar (שחייב להיות בשם ex2.tar) ולספק את הנתיב שלו לפריסבמיט, כך:

```
~labcc/presubmit/ex2/run <path_to_ex2.tar>
```

בהצלחה😊😊