20.05.20

דגשים והבהרות לתרגיל 3 ב-C

הערה: קובץ זה אינו קובץ רשמי של הקורס, ונעשה ע"י סטודנטים בלבד.
איננו יכולים להבטיח שלא נפלה טעות באחת ההבהרות, ולא לוקחים אחריות. נא להפעיל גם את שיקול דעתכם :)

קבצי header:

שימו לב שכל אחד משני קבצי ה-header "מחולק" ל-3:

1. Structs - הכרזה על הסטראקטים והמשתנים שלהם בהם נשתמש.
2. Typedefs שהובאו לנוחיותנו - עבור למשל פוינטרים לפונקציות מסוימות וכד'.
3. הכרזה על פונקציות.

לגבי structs ו- typedefs אין צורך לעשות כלום.
כל המימושים שעלינו לעשות - הם אך ורק על הפונקציות המוכרזות שם.

פונקציות כלליות:

מלבד הפונקציות שחובה עלינו לממש בדיוק לפי הדרישה - מותר ואף רצוי לממש כל פונקציה אחרת שנרצה.
במקרה הזה אין צורך להוסיף הכרזה שלה לקובץ ההדר, היא פונקציה פרטית לשימוש שלנו ואין צורך שהמשתמש ידע עליה (אפשר לחשוב על זה קצת כמו על API).
פונקציות כלליות שמומלץ לממש:
Successor, findMin, leftRotation, rightRotation

חלק א' - RBTree.c

הוספה לעץ:

- תיקון העץ לאחר הוספת איבר:

הסעיף הרביעי כתוב בצורה לא כ"כ ברורה.
דבר ראשון - תמיד נעשה גם את 4ב' וגם את 4ג', פשוט לפני כן צריך לבדוק האם א' התקיים.
דבר שני - במקרה שא' התקיים נחליף את היצוג אצלנו בין N ל-P לעניין הצבעים.
זאת אומרת שאח"כ כשאנחנו משנים את הצבעים מי שמשנה את הצבע לשחור יהיה N בעצמו ולא האבא.
לעומת זאת אם דילגנו על סעיף א' אז כן האבא הוא זה שישנה את הצבע שלו.
לגבי הסבא אין שינוי, תמיד ישנה את צבעו, בין אם סעיף א' התקיים או לא.

- רקורסיה בתיקון העץ:

הסעיף היחיד שיכול לשלוח לרקורסיה הוא סעיף 3, אבל הקריאה הרקורסיבית היא על כל האפשרויות של ההכנסה.
מאפשרות מספר אחד - שזה השורש ... וכו'.

20.05.20

מחיקה בעץ:

- את מי מוחקים?

נסמן ב-M את הקודקוד אותו אנחנו רוצים למחוק.
  - אם אין ל-M שני ילדים אמיתיים (ראו ערך: ילד אמיתי), כלומר ילדים שהם לא הפניות NULL - נדאג למחוק את M בעצמו כמו במחיקה רגילה בעץ בינארי:
    נסדר את המצביעים של הילד שלו (אם קיים) ושל ההורה ש"ילדגו" מעליו. ולאחר מכן נמחק ממש את הקודקוד.
  - אם ל-M יש שני ילדים אמיתיים - נמצא את העוקב (successor) של M, נחליף בין הערכים שלהם, ומעכשיו כשמדברים על M במחיקה הכוונה למיקום של היורש.
    רק אם ל-M יש שני ילדים נמצא את הסוקסור שלו, ובפרט מובטח לנו שיהיה קיים כזה, שהרי שני ילדים -> בן ימני קיים -> יש קודקוד שגדול ממנו.

בין אם יש ל-M ילד אחד, שניים, או שאין בכלל - תמיד נשלח לאחר המחיקה לתיקון העץ!

- המחיקה עצמה:

כשמוחקים קודקוד מהעץ צריך לשחרר גם את הדאטה שלו.
כלומר מלבד שחרור הזיכרון של הקודקוד עצמו צריך להשתמש גם בפונקציית השחרור שניתנה לנו עם יצירת העץ ע"מ לשחרר ספציפית את הדאטה של אותו קודקוד.
הדאטה זה השדה היחיד בקודקוד שנשחרר מלבד הקודקוד עצמו.

- תיקון המחיקה:
  - *סימון ה-DB:*

זה סימון **סמנטי בלבד**. אין לו ייצוג בקוד ואין צורך בייצוג כלשהו.
פשוט את הקודקוד שמפר את המצב בעץ, לפי התנאים שבדקנו, נשלח רקורסיבית לפונקציה שמתקנת את ה-DB.

  - *טיפ שלי:*

בשלב המחיקה כשנכנסים למצב של DB (דהיינו מחקנו קודקוד שחור וצריך לדאוג לאזן בחזרה את כמות הקודקודים השחורים בעץ) יש אפשרות לתקן את העץ שוב ושוב ברקורסיה.
לגמרי יכול להיות מצב שהקודקוד ה"בעייתי" הוא בכלל עלה ואין לנו יצוג אמיתי שלו בתור קודקוד.
אני הייתי ממליצה דבר ראשון למחוק את הקודקוד שצריך בפונקציה נפרדת, ולפונקציית התיקון הרקורסיבית לא לשלוח רק את הקודקוד עצמו, אלא להתחשב בעובדה שהוא יכול להיות כלום ושום דבר ולשלוח למשל את ההורה שלו (לפי ההורה אפשר להסיק המון דברים).

- רקורסיות:

רקורסיה בתיקון המחיקה יכולה להיות רק מתוך סעיף 3, ונקראת רק על סעיף 3 - כלומר אנחנו לא נכנס איכשהו למקרה 1 או למקרה 2 עבור המחיקה, אלא רק לתתי הסעיפים של מקרה 3.
(בעצם כל סעיף 3 הוא הטיפול המעצבן ב DB, וכשאנחנו קוראים רקורסיבית זה כי נשארה בעיה עם DB).
בתוך הסעיף הזה קוראים רקורסיבית רק בתתי האפשרויות: ב2, ג, ד
וכשקוראים רקורסיבית קוראים לפונקציה שתבדוק את כל המקרים בסעיף 3 - מא' עד ה'.

20.05.20

חלק ב' - Structs.c

פונקציית copyIfNormIsLarger

הוקטור שמעתיקים אליו הוא אישיות בפני עצמה.
הוא לא העתק של וקטור קיים, ובפרט לא נשנה אותו "להצביע" על וקטור קיים.
ההעתקה צריכה להיות deep copy
כלומר נרצה להעתיק לגמרי את שני השדות של הווקטור עם הנורמה הגדולה:
את הגודל שלו
ואת המערך שמייצג את הווקטור.
ואכן את העתקת המערך נעשה באמצעות הקצאת זיכרון (עדיף ריאלוק תמיד, שכן זה גם מקצה מחדש אם לא קיים זיכרון, וגם משנה את הזיכרון בהתאם למה שצריך :)).

פונקציות לחלק של המחרוזות

פונקציות ספריה שימושיות שמותר ואף מומלץ להשתמש בהן: strcpy, strcat

טעויות נפוצות

(שלא תרצו להיות חלק בסטטיסטיקה שלהן... :))

- לאתחל, לאתחל, לאתחל, ושוב - לאתחל!
  שימוש ב malloc לשם יצירת קודקוד ללא אתחול ההפניות השונות שלו עלול לשים ערך זבל באחד המצביעים, וכשנתקדם למשל במורד העץ נגיע למקומות בזיכרון שלא נרצה להגיע אליהם!
  או שתשתמשו ב calloc או שתדאגו לאתחל הכל מיד לאחר ההקצאה.

- כשרוצים לשנות את שורש העץ - לא לשלוח את הקודקוד שכרגע מהווה שורש! לשלוח את העץ עצמו, ודרך השדה root לגשת לשורש... אחרת הפעולה שלכם לא באמת עשתה משהו.

- בשימוש בפונקציות שניתנו לנו עם יצירת העץ - שימו לב לשלוח את ה **data** של הקודקוד, **ולא את הקודקוד עצמו**!

20.05.20

הערות כלליות:

- האלגוריתם בנוי על לא מעט דקויות. וודאו שאתם מבינים מה לעשות בכל פונקציה אותה אתם מתחילים לבנות. חבל על הזמן שיתבזבז אח"כ בניסיון להבין מהו המצביע המסוים שהתבלבלתם במה שצריך לקרות איתו, וכו'...
- גלגולים בעץ:
  אם הנושא לא יושב טוב יש לא מעט מקורות באינטרנט שמבהירים בצורה מסודרת.
  טכנית כשמבצעים גלגול על קודקודים - בד"כ לפונקציה שמגלגלת שולחים את האבא.
- "רוטציה" -
  כשהם אומרים זאת (למשל בהכנסה ב4א') הכוונה לגלגול בעץ, ולא סתם להחלפה בין קודקודים.
- אתר מעולה להדמיית עץ אדום-שחור: אתם יכולים להוסיף/למחוק איברים מהעץ כאוות נפשכם ולראות את התהליך שמתבצע בעץ בזמן ההוספה/המחיקה. מומלץ מאוד! יכול לעזור להעניק אינטואיציה אפילו לפני שהתחלתם ממש לכתוב את התרגיל בעצמכם.