תרגיל בית 3

בתרגיל בית זה נממש בשפת C מבנה נתונים השומר נקודות במישור. לכל נקודה קואורדינטת x וקואורדינטת y. הפעולות שעלינו לממש:

- Insert(x,y): הכנס את הנקודה (x,y) למבנה.
- Delete(x,y): הוצא את הנקודה (x,y) מהמבנה.
- Delete_third(x_1,x_2): מחזירה את מספר הנקודות שקואורדינטת הx שלהם היא בין x_1 ל x_2 ואז מוחקת שליש מהם. פורמלית, נגדיר:

$$A = \{ (x, y) \mid x_1 \leq x \leq x_2 \}$$

נגדיר את שליש האיברים האמצעי של A בתור W כאשר:

$$A = U \cup W \cup Z,$$

$$|U| = |W| = |Z| = \frac{|A|}{3},$$

$$\forall\, u \in U, w \in W, z \in Z: u < w < z$$

על הפעולה להחזיר את הערך $|A|$, ואז למחוק את כל האיברים ב W. **שימו לב: ניתן להניח שהגודל של A תמיד יתחלק ב-3.**

ניתן להניח שאין שתי נקודות שונות עם אותו ערכי x

מימוש עץ AVL:

מצורפים שני קבצים:

- avl.c
- avl.h

הקובץ avl.h מכיל את הגדרת העץ, ואין לשנותו.

הקובץ avl.c מכיל כותרות פונקציות ריקות שעליכם לממש. שימו לב: **לא ניתן לשנות את הכותרות.**

הערות כלליות:

- הנקודות שעליכם לשמור מכילות (x,y), ויש לשמור את שני הערכים של כל נקודה בצומת העץ. על העץ להיות ממויין לפי ערכי x (כלומר מפתחות העץ הם ערכי x).
- אין צורך לשמור שתי נקודות עם אותו ערך x, תצטרכו לבדוק זאת בעת ההכנסה.
- ניתן להניח שהקלט תמיד תקין, כלומר לא תקבלו ערכים לא מתאימים בתור ארגומנטים באף שלב.

הפונקציות:

- AVLNodePtr avl_search(AVLNodePtr root, int x , int y): הפונקציה מקבלת שורש לעץ, ושני ערכים x,y. הפונקציה מחפשת את הנקודה (x,y), ומחזירה מצביע אליו אם הוא נמצא. אחרת מחזירה NULL
- AVLNodePtr avl_insert(AVLNodePtr root, int x, int y): הפונקציה מקבלת נקודה (x,y) ומוסיפה אותה לעץ. אם כבר ישנה צומת עם מפתח x, לא צריך לעשות כלום. הפונקציה מחזירה מצביע לשורש העץ.
- AVLNodePtr avl_delete AVLNodePtr root, int x, int y): הפונקציה מקבלת נקודה (x,y) ומוחקת את הנקודה הזו מהעץ אם היא קיימת. הפונקציה מחזירה מצביע לשורש העץ.
- int delete_third(AVLNodePtr root, int x1, int x2): הפונקציה מקבלת שני ערכים $x_1 < x_2$. הפונקציה מחזירה את מספר הנקודות שקואורדינטת האיקס שלהם בתחום $[x_1, x_2]$, ואז מוחקת את השליש האמצעי של הנקודות בתחום הנ"ל.
- AVLNodePtr new_avl_node(int x, int y): הפונקציה מקבלת שני ערכים ומחזירה צומת שערכה (x,y). אם הקצאת הזיכרון נכשלת יש להחזיר NULL
- void delete_avl_tree(AVLNodePtr root): הפונקציה מקבלת מצביע לשורש העץ, ומוחקת את כל הצמתים מהזיכרון (free).

void submitters(): פונקציה שמדפיסה את שמות המגישים, כולל ת.ז, לפי הפורמט הבא:

Name1
ID1
Name2
ID2

קומפילציה:

הקומפיילר צריך להיות GCC עם הדגלים הבאים:

- -Wall
- -Wextra
- -pedantic-errors

ניקוד:

- חיפוש 20
- הכנסה 20
- מחיקה 20
- Range_query 40

הנחיות הגשה והערות:

- אין להשתמש בספריות נוספות מלבד stdio,stdlib.
- אין לשנות את הקובץ avl.h.
- מותר לממש פונקציות עזר כרצונכם בקובץ avl.c.
- אין לשנות את ערך המפתח של צומת אחרי שכבר יצרנו אותו.
- הקוד שהוגש צריך להיות מתועד ונקי.
- **אין להוסיף main לavl.c.**
- **שימו לב שע"מ לבדוק את אמינות הקוד, נשתמש בכלי לבדיקת העתקות (גם מהאינטרנט וגם אחד מהשני)**
- **יש להגיש אך ורק את הקובץ avl.c עם אותו השם. ההרצה מתבצעת באופן אוטומטי ולכן תרגילים שלא יתקמפלו או עם שמות שונים יקבלו 0.**

טיפים:

- ממשו קודם את הפונקציות הבסיסיות.
- השתמשו ברקורסיה.
- אל תחסכו בפונקציות עזר.
- מומלץ לתכנן בדיקות משלכם.

בונוסים:

ל-3 התרגילים שירוצו הכי מהר ינתן בונוס לציון הסופי, בהדרגה.

בהצלחה.