החוג למדעי המחשב
המכללה האקדמית תל-חי

מבוא לתכנות מערכות
תרגיל בית מספר 5

נושא: פוינטרים לפונקציות
(ועוד תוספות....)

סמסטר אביב 2015-16

תאריך הגשה: 13.06.2016, שעה: 23:55
הגשה בזוגות

בהצלחה!

חלק 1 (ADT – Abstract Data Type)

בשאלה זו נציג ADT בשם WEI המכיל עצמים משני סוגים

1. מספרים (ממשיים).
2. נקודות (במישור).

WEI פועל כדלקמן:

- כל מספר מזוהה ע"י עצמו. לדוגמה: 3.14159
- המזהה של נקודה הוא מחרוזת תוים (לפי בחירת המשתמש). כדי להכניס נקודה ל-WEI יש לנקוב בשני מספרים (החייבים כבר להימצא בתוך ה-!): WEIהראשון ישמש כקואורדינטת X של הנקודה, והשני ישמש כקואורדינטת Yשל הנקודה.

ממשו את קובץ המנשק של המודול WEI. בנוסף ל-type-ים וכד', על המנשק לתמוך בפונקציות הבאות בלבד:

1. יצירת WEI.
2. הוספת מספר ל-WEI.
3. הוספת נקודה ל-WEI.
4. החזרת אינדיקציה האם שלוש נקודות ב-WEI קולינאריות (נמצאות על קו ישר) או לא.
5. בהינתן שתי נקודות ב-WEI, החזרת שיפוע הקו העובר דרכן (טנגנס הזוית שיוצר הקו עם הכיוון החיובי של ציר X).
6. בהינתן מספר k ב-WEI, החזרת כל הנקודות אשר k הוא קואורדינטת X שלהן או קואורדינטת Y שלהן. (הכוונה ב-"או" היא שהמשתמש בוחר אחת מן האפשרויות).
7. הריסת WEI.

בשאלה זאת יש להגיש קובץ WEI.h בלבד!

חלק 2 (Bash)

כתוב תסריט המבצע החלפת סדר המילים במשפטים המופיעים בקבצי טקסט נתונים (כל קבצי הטקסט בתיקיה מסוימת). כל קבצי הטקסט מסתיימים ב- "tex."

קלט התוכנית

- כל קבצי הטקסט בתיקיה.
כל קובץ טקסט מכיל משפטים - אוסף המילים המופרדות ברווחים.
כל משפט מסתיים בסימן נקודה.
כל שורה בקובץ יכולה להכיל יותר ממשפט אחד.
משפט לא יכול להתפרס על יותר משורה אחת.

פלט התוכנית

לכל קובץ קלט יש ליצור קובץ חדש ששמו כשמו של הקובץ המקורי בתוספת הסיומת ".rpl" המכיל את החלפת סדר המילים במשפטים המופיעים בקובץ טקסט נתון. כל משפט בקובץ החדש יתחיל משורה חדשה.
למשל, המשפט I love Matam Course. מהקובץ טקסט הנתון יוחלף במשפט Course Matam love I.

חלק 3 (המשך תרגיל בית 4)

מטרת התרגיל

עבודה עם פוינטרים לפונקציות.
העבודה הנוכחית תהווה הרחבה של תרגיל בית 4 יחד עם השינויים הדרושים לשימוש **בעצים כלליים**.

תאור התרגיל

בנק " החסכוני הראשון" החליט שוב לשפר את התוכנה. התכנה הנוכחית מתוכננת להיות הרבה יותר קצרה מהתכנה הקודמת מבחינת כמות הקוד הנדרש למימוש.

שלב 1

עליך לממש את הפעולות הבאות **לנתון כללי**, מבלי להניח אילו פרטים תצטרך להגדיר עבור אותו הנתון .אתה מתבקש להשתמש במבנה נתונים מסוג **עץ חיפוש בינארי** לשם כך:

- הגדרת עץ **כללי** ריק **CREATE_TREE** **ה**מגדיר עץ חיפוש בינארי ריק של נתונים כלליים.
- הוספת נתון חדש למערכת: **ADD_NEW_NODE** המקבלת מהמשתמש פרמטרים של הנתון **הכללי** ומוסיפה נתון זה למערכת
- מחיקת נתון מהמערכת: **REMOVE_NODE** המקבלת נתון ומורידה אותו מן המערכת.
- **FIND_NODE** - מציאת נתון במערכת. אם יש יותר מאחד, יש למצוא את כולם ולבנות מהם רשימה מקושרת.
- **AVERAGE_KEY** - חישוב ממוצע של כלל הנתונים במערכת
  יש לבצע פעולה זאת בצורה רקורסיבית

6. **TREE_TO_ARRAY**
הפונקציה מקבלת את עץ ה נתונים ובונה ממנו בצורה רקורסיבית מערך נתונים.

7 **PRINT_LIST** – הדפסת רשימת נתונים כלליים עם כל פרטיהם.
8 . **FREE_LIST** – שחרור מבנה נתונים

שלב 2.

א). **השתמש בפונקציות שהגדרת בשלב 1**, על מנת לנהל בסיס נתונים עבור הסניפים והלקוחות כפי שהיה נדרש בתרגיל בית 4.
(**שים לב:** עליך לצמצם בצורה מרבית את הקוד שיצרת בתרגל בית מספר 4!)

אתה מתבקש לשפר את המערכת באופן שיתואר להלן.
המערכת תכלול ניהול של 3 דברים: ניהול הבנק, ניהול סניפים, ניהול לקוחות.

- הבנק מאופיין ע"י 6 שדות:
- שם (שם יהיה מורכב ממילה אחת באורך *לא ידוע*)
- מספר הסניפים (מספר מ 1 עד 500)
- מספר חשבונות בבנק
- סכום הכסף של כל הלקוחות יחד

- רווחים נטו של השנה האחרונה
- מספר הלוואות פעילות שניתנו ע"י הבנק ללקוחות

- כל **סניף** מאופיין ע"י 9 שדות:
- שם הבנק אליו שייך הסניף (באורך לא ידוע)
- מספר סידורי של הסניף (מספר בין 1 ל 500)
- שם הסניף (באורך לא ידוע)
- מספר חשבונות בסניף
- סכום הכסף של כל לקוחות הסניף (מספר הנמדד במיליונים – בין 1 לבין 10000)
- רווחים נטו של השנה האחרונה בסניף (מספר הנמדד במיליונים – בין 1 לבין 10000)
- מספר הלוואות פעילות שניתנו ע"י הסניף ללקוחות
- שעת פתיחת הסניף (ניתן להניח כי השעה היא שעה עגולה ושבכל ימי השבוע הסניף נפתח באותה השעה)
- שעת סגירת הסניף (ניתן להניח כי השעה היא שעה עגולה ושבכל ימי השבוע הסניף נסגר באותה השעה)

- כל **לקוח** מאופיין ע"י 10 שדות: - שם פרטי (*באורך לא ידוע*)
- שם משפחה (*באורך לא ידוע*)
1. ת.ז. (מספר שלם בן 9 ספרות)
- שם הבנק בו מתנהל החשבון (באורך לא ידוע)
  - מספר סניף בו מתנהל החשבון
  - מספר חשבון (אם ללקוח יש מספר חשבונות, כל אחד ייחשב כלקוח אחר) (מספר שלם בן 5 ספרות)
  - חריגה מורשית - סכום בש"ח של המינוס המותר ללקוח (מספר מ 0 עד 200000 ש"ח)
  - יתרה בחשבון עו"ש
  - יתרה בהלוואות לבנק - כמה כסף חייב הלקוח לבנק בגלל ההלוואות שנלקחו על ידו.
  - יתרה בחשבון בתוכניות פק"ם (כאשר הכסף סגור לתקופות שונות)

<u>עליך לממש של הפעולות הבאות:</u>

1. הגדרת עץ הסניפים **createBranchTree** המגדירה **<u>עץ חיפוש בינארי ריק</u>** של סניפים.

- הוספת סניף חדש למערכת: **addNewBranch** המקבלת מהמשתמש את כל הפרמטרים הדרושים ומגדירה סניף חדש במערכת.

- הגדרת עץ לקוחות הסניף **createBranchClientTree** המגדירה **עץ חיפוש בינארי ריק** _ של לקוחות הסניף.

- הוספת לקוח חדש לסניף במערכת: **addNewClientToBranch** המקבלת מהמשתמש את כל הפרמטרים הדרושים של הלקוח ומספר הסניף אליו הצטרף הלקוח ומגדירה לקוח חדש במערכת.
- הפקדת כסף ע"י לקוח לחשבון עו"ש שלו **depositeMoneyToClientAccount** המקבלת מהמשתמש את כל הפרמטרים הדרושים של הלקוח וכמות הכסף המופקדת ומעדכנת את יתרת העו"ש של הלקוח.
- לקיחת הלוואה מהבנק ע"י הלקוח **loanToClient** המקבלת מהמשתמש את כל הפרמטרים הדרושים של הלקוח וכמות הכסף המולוות ומבצעת את ההלוואה.

- שאילתא של מספר הלקוחות של הבנק **clientNumberOfBank** המחזירה את מספר החשבונות המנוהלים ע"י הבנק.

- שאילתא של מספר לקוחות הסניף בעלי יתרת עו"ש גבוהה מ סכום נתון: **clientNumberWithGivenBalance** המקבלת מספר בש"ח המהווה יתרה לבדיקה ומחזירה את מספר הלקוחות בעלי יתרת עו"ש הגבוהה מהמספר הנתון.
- שאילתא של מספר לקוחות הסניף בעלי יתרת ההלואות הגבוהה מיתרת עו"ש שלהם :**clientNumberWithBiggerLoansThanBalance.**

- מציאת לקוח **findClientInGivenBranch** לפי אחד משני פרמטרים: ת.ז. או יתרה בחשבון עו"ש. אם יש יותר מלקוח אחד, יש למצוא את כולם ולבנות מהם רשימה מקושרת של הממוינת לפי מספר ת.ז.

- שאילתא של ממוצע של מספר חשבונות הבנק המנוהלים ע"י הסניפים **averageNumberOfAccountsInBranches** המחשבת את מספר החשבונות, במומוצע, המנוהלים ע"י כל הסניפים של הבנק.

- הדפסת מספרי החשבון של לקוחות של סניף נתון יחד עם יתרתם **printClientAccountsNumberAndBalance** המקבלת את מספר הסניף של הבנק ומדפיסה את הדרוש.
- מחיקת לקוח מהמערכת **deleteClient** המקבלת מהמשתמש את מספר החשבון של הלקוח ומוחקת אותו מהמערכת.
- מחיקת **כל** לקוחות הסניף מהמערכת **deleteAllBranchClients** המקבלת את מספר הסניף ומוחקת את כל לקוחותיו מהמערכת.
- מחיקת סניף של הבנק מהמערכת **deleteBranch** המקבלת מהמשתמש את מספר הסניף ומוחקת אותו מהמערכת.
- מחיקת **כל** הסניפים מהמערכת **deleteAllBranches** המוחקת את כל הסניפים מהמערכת.

דגשים:

- בכל הפונקציות הרקורסיביות אתם מתבקשים לא להשתמש במשתנים סטטיים ופונקציות עזר.
- **יש לבדוק את תקינות הקלט לכל הפונקציות. במקרה שהקלט לא תקין, יש להציג הודעה על שגיאה.**
- בדוק שהנך מטפל גם במקרי קצה.
- **יש לשמור על קונסיסטנטיות בין המבנים!**

- יש לתכנן היטב את פתרון התרגיל טרם תחילתו. יש להקפיד על התיכנון הנכון וחלוקת המשימות לקבצים
- יש להגיש תוכנית המכילה קבצי מקור וקבצי header וכן פונקציה ראשית main המדגימה את הבדיקות שנעשו לתוכנה שנכתבה.

הודעות שגיאה

סוגי השגיאות עליהן יש לדווח.

- קלט לא תקין

הידור, קישור ובדיקה עצמית

- יש לקמפל ולהריץ את התוכנית ב LINUX. שים לב: תוכנית שלא תתקמפל במערכת הפעלה LINUX תקבל ציון 0!

2. בקומפילציה יש להיעזר בדגלים -Wall –pedantic-errors -ansi, אשר עוזרים לשפר את איכות הקוד. שימו לב שכאשר משתמשים בדגל -pedantic-errors הקומפיילר מחשיב את ה-warnings כ-errors! **הבדיקה תיעשה בעזרת הדגלים הללו, ולכן על התוכניות שלכן להתקמפל ללא warnings.**
כמו כן, בקומפילציה יש להיעזר בדגל -lm, שיכלול בתוכניתכם את קובץ הספרייה math.h, (אם יהי צורך בכך) ע"מ שתוכלו להשתמש בפונקציות של הספריה .

דרישות, הגבלות הערות רמזים ותוספות:

1. יש להקפיד על תכנון נכון של התוכנית וכתיבה נכונה ב-C. בשלב זה של לימודיכם הנכם מסוגלים ונדרשים להיות מסוגלים לתכנן ולכתוב תוכנית בסדר הגודל הנתון בעצמכם. תכנון התוכנית מהווה חלק מהותי מהתרגיל. הקפידו לבצע חלוקה של התוכנית למודולים ופונקציות בהתאם למבנה הלוגי של התוכנית ולא לכתוב את התוכנית כפונקצית main אחת גדולה. לדוגמא, נצפה לראות הפרדה בין הפונקציות המטפלות בקלט, לבין אילו המנהלות את מבני הנתונים השונים.

2. יש לתעד את התוכנית. את אופן התיעוד אנו מניחים לכם לקבוע בעצמכם. ההנחיה היחידה שתינתן הנה כי מי שרוצה להיעזר בפונקציות שאתם כותבים, צריך להיות מסוגל לבצע זאת ע"י הסתכלות בהצהרת הפונקציה וקריאת התיעוד ללא צורך בקריאת גוף הפונקציה.

הגשת התוכנית:

עליך להגיש קובץ מכווץ (zipped file) שבו הקבצים המכילים את תוכניתכם (אותם אתם כתבתם)
ניתן ליצור את הקובץ ה-zip ב-2 הדרכים הבאות (לדוגמא):

1. ב-windows: בעזרת WinZip.
2. ב-linux: בעזרת הפקודה zip. לדוגמא:

zip zip_file_name file_1 file_2 ... file_n.

שימו לב:

על התרגיל להיות מוגש בזוגות. הנכם רשאים להגיש לבד, אך הדבר אינו מומלץ. עומס התרגיל תוכנן עבור שני סטודנטים. הגשה לבד לא תעניק הקלות.

הגשה באיחור תגרור קבלת ציון 0 בתרגיל

The errors:

A: Program was not split into separate modules / it was splited badly.
B: too long functions (long function bodies). Using long functions makes the code
hard to read, understand and maintain. A well written function shall have a
clearly defined task and shall perform only that task, not several tasks at the same time.
C: unnecessary "include"
D: didn't check return arguments of the given functions.
E: use of global variable.
F: Bad names of variables/functions (not meaningful, limited in length, not in English, etc.)
G: No or bad documentation (comments)/ the code is unreadable...
H: duplication of code.

I: Rare and generally serious errors