החוג למדעי המחשב
המכללה האקדמית תל-חי

מבוא לתכנות מערכות
תרגיל בית מספר 5

נושאים: פוינטרים לפונקציות.

סמסטר אביב 2017-18

תאריך הגשה: 19/06/2018, שעה: 23:55
הגשה בזוגות

בהצלחה!

תרגיל בית נוכחי מורכב משני חלקים המהווים חלקים בלתי נפרדים. החלק הראשון עוסק ב ADT והשני מהווה הרחבה של תרגיל בית 3 שהגשתם לפוינטרים לפונקציות.

חלק 1 (ADT)

שאלה 1

בשאלה זו נממש ADT בשם JobInquiry התומך בשאילתות של משתמש לקבלת הצעות עבודה. כל הצעת עבודה מכילה *שם החברה, שם המשרה ומשכורת*.

המשתמש יכול (אך אינו חייב) לקבוע את הפרמטרים הבאים:

- מספר הצעות שהוא רוצה לקבל
- החברה בה הוא מחפש עבודה
- המשרה שהוא מחפש
- המשכורת המינימאלית לה הוא מצפה

במקרה שפרמטר כלשהו לא נקבע, אין מגבלה על הפרמטר הזה. כלומר, שאילתא תחזיר כל הצעות העבודה העונות
על הדרישות של כל הפרמטרים שנקבעו ע"י המשתמש.

הערה:

כל הצעות העבודה מכל החברות רשומות בקובץ בודד לפי סדר כרונולוגי (הצעות מאוחרות יותר נמצאות מאוחר יותר
בקובץ). שם הקובץ ומספר הצעות בקובץ מועברים כפרמטרים בזמן היצירה של JobInquiry.

ממש/י את **קובץ המנשק** (אין צורך לממש פונקציות עצמן!) של המודול JobInquiry.בנוסף לכל הטיפוסים הדרושים, על המנשק לתמוך בפונקציות הבאות בלבד:

1. jobInquiryCreate – יצירת JobInquiry.

2. jobInquiryGetHotJob – הדפסת המשרה הכי "חמה" (אחרונה בסדר כרונולוגי) שנמצאת במבנה הנתונים (לא תלויה בפרמטרים שהגדיר או לא הגדיר המשתמש).

3. jobInquirySetNumber – קביעת מספר המשרות הרצויות(מספר גדול מ-0).

4. jobInquirySetPosition – קביעת משרה רצויה.

5. jobInquirySetSalary – קביעת משכורת מינימלית (מספר לא שלילי).

6. jobInquirySetCompany – קביעת חברה רצויה.

7. jobInquiryPrintAndDestroy – הדפסת כל המשרות הרצויות בסדר כרונולוגי הפוך (הצעות המאוחרות יותר יודפסו מוקדם יותר) עד למקסימום הצעות הנקבע ע"י המשתמש או עד שאין יותר הצעות מתאימות אם משתמש לא קבע פרמטר זה. לאחר הדפסה המודול נהרס.

8. jobInquiryDestroy – הריסה של מודול ללא הדפסה.

שאלה 2

בנמל התעופה "בית נחיתון" יש N מסלולים. כל אחד מהמסלולים משמש גם להמראה וגם לנחיתה.
מגדל הפיקוח של נמל התעופה מנהל את ההמראות והנחיתות ונותן למטוסים הרשאות להמראה ונחיתה.

מטוס המבקש להמריא, שולח למגדל הפיקוח בקשה להמראה, הכוללת את מספר המטוס (מספר בן 4 ספרות).
מטוס המבקש לנחות שולח למגדל הפיקוח את מספר המטוס ואת השעה (ללא הדקות) שבה ינחת בנמל התעופה.

מגדל הפיקוח נותן למטוסים אישורים להמראה על פי סדר הגעת הבקשות.
מגדל הפיקוח נותן למטוסים אישורים לנחיתה לכל מטוס ששלח בקשה לנחיתה, לשעה שבה הוא מבקש לנחות.
הפיקוח נותן עדיפות למטוסים המבקשים לנחות, על פני מטוסים המבקשים להמריא.

בסוף ההמראה או הנחיתה, הטייס מודיע למגדל הפיקוח על פינוי המסלול.

הנחה:

- לא יתכן מצב שמטוס מגיע לנחיתה ואין אף מסלול פנוי.
- לעומת זאת, יתכן מצב שמטוס המבקש להמריא נאלץ להמתין למסלול פנוי.

ברצונינו להגדיר ADT "פיקוח נמל תעופה" המנהל את מערך הטיסות.

א). יצג את המידע הנדרש לניהול ההמראות והנחיתות (בסעיף זה יש להגדיר את כל סוגי המבנים הדרושים למשימה).

ב). כתוב **מנשק** לטיפוס הנתונים "פיקוח נמל תעופה". הגדר ותאר את הפעולות והפרמטרים.

שים לב: אין צירך במימוש הפעולות הנ"ל

חלק 2 (פוינטרים לפונקציות)

מטרת התרגיל

עבודה עם פוינטרים לפונקציות.
העבודה הנוכחית תהווה הרחבה של תרגיל בית 3 יחד עם השינויים הדרושים לשימוש **בעצים כלליים**.

תאור התרגיל

אתם מתבקשים שוב לשפר תוכנה ישנה לניהול גני החיות אשר נמסרה לפני זמן קצר. התוכנה הבאה 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 **CreateTree** המגדיר עץ חיפוש בינארי ריק של נתונים כלליים.

- הוספת נתון חדש למערכת: **AddNewNode** המקבלת מהמשתמש פרמטרים של הנתון **הכללי** ומוסיפה נתון זה למערכת

- מחיקת נתון מהמערכת: **RemoveNode** המקבלת נתון ומורידה אותו מן המערכת.

- **FindNode** - מציאת נתון במערכת. אם יש יותר מאחד, יש למצוא את כולם ולבנות מהם רשימה מקושרת.

- **AverageKey** - חישוב ממוצע של כל הנתונים במערכת *יש לבצע פעולה זאת בצורה רקורסיבית*

- **TreeToArray** הפונקציה מקבלת את עץ הנתונים ובונה ממנו בצורה רקורסיבית מערך נתונים.

- הדפסת את כל הנתונים הכלליים עם כל פרטיהם של העץ **PrintTree.**

- שחרור מבנה נתונים **FreeTree.**

שלב 2.

א). **השתמש בפונקציות שהגדרת בשלב 1**, על מנת לנהל בסיס נתונים עבור הלקוחות, המכוניות והספקים כפי שהיה נדרש בתרגיל הבית 3.
(**שים לב:** עליך לצמצם בצורה מרבית את הקוד שיצרת בתרגל בית מספר 3!)

אתם מתבקשים לשפר את המערכת באופן שיתואר להלן.

- המערכת תכלול ניהול של 3 דברים: ניהול חיות הנמצאות בגן, ניהול העובדים המטפלים בחיות וניהול של סוגי האוכל הניתנים לחיות.
-
- כל **חיה** מאופיינת ע"י 11 תכונות:
- סוג (שם יהיה מורכב ממילה אחת באורך ***לא ידוע***)
- צבע (באורך ***לא ידוע***)
- שם (באורך ***לא ידוע***)
- מספר רישומי (מספר איחודי בן 5 ספרות)
- תאריך הלידה (בפורמט: dd/mm/yyyy)
- מקום לידה (באורך ***לא ידוע***)
- מצב בריאותי (מספר שלם בין 1 ל 100)
- מספר ילדים (מספר שלם מ 0 עד 1000)
- מספרי רישום של ילדיו (מערך בגודל 1000 בו מופיעים מספרי הרישום של ילדיו)
- סוג אוכל (מספר איחודי שלם בן 20 ספרות = הבר קוד)
- מספר ארוחות ביום (מספר שלם מ 0 עד 10)
-
- כל **מטפל** מאופיין ע"י 6 תכונות:
- - שם פרטי (באורך ***לא ידוע***)
- - שם משפחה (באורך ***לא ידוע***)
- ת.ז. (מספר שלם בן 9 ספרות)
- מספר החיות הנמצאות בטיפולו (מספר שלם בין 0 ל 200)
- מספרי הרישום של החיות בהן מטפל העובד (מערך בגודל לא ידוע בו מופיעים מספרי הרישום של החיות)
- - תאריך תחילת העבודה (בפורמט: dd/mm/yyyy)
-
- כל **סוג אוכל** מאופיין ע"י 5 תכונות:
- - מספר בר-קוד של האוכל (מספר איחודי שלם בן 20 ספרות)
- - שם החברה המייצר (באורך ***לא ידוע***)
- - סוג חיות הצורכות את האוכל (מערך של סוגי החיות בגודל לא ידוע)
- מספר השקים הנמצא במלאי (מספר שלם מ 0 עד 10000)
- גודל של כל שק (מספר לא שלם המסמן מספר הק"ג, מ 1 עד 50)
-
- **עליך לממש את הביצוע של הפעולות הבאות:**

1. הגדרת עץ החיות **createAnimalList** המגדירה ***עץ ריק*** של חיות.
2. הוספת חיה חדשה למערכת: **addNewAnimal** המקבלת מהמשתמש את כל הפרמטרים הדרושים ומגדירה חיה חדשה במערכת.
3. הגדרת עץ מטפלי גן החיות **createEmployeeList** המגדירה ***עץ ריק*** של המטפלים.
4. הוספת מטפל חדש למערכת: **addNewEmployee** המקבלת מהמשתמש את כל הפרמטרים הדרושים של המטפל ומגדירה מטפל חדש במערכת.
5. הגדרת עץ סוגי האוכל **createFoodList** המגדירה ***עץ ריק*** של סוגי האוכל.
6. הוספת סוג אוכל חדש למערכת: **addNewFood** המקבלת מהמשתמש את כל הפרמטרים הדרושים של סוג האוכל ומגדירה סוג אוכל חדש במערכת.

7. שאילתא של מספר החיות הצורכות סוג אוכל מסוים הנתון ע"י בר-קוד מסוים **animalNumberWithGivenFoodKind** המחזירה את מספר החיות האוכלות אוכל מאותו הסוג (עם אותו הבר-קוד).
8. שאילתא של מספר החיות בני אותו צבע :**animalNumberWithGivenColor** המקבלת שם של צבע ומחזירה את מספר החיות בגן החיות בעלי הצבע הנ"ל.
9. שאילתא של 3 סוגי האוכל הנצרכים ביותר ע"י החיות :**threePopularFoods** המחזירה את מספרי הבר קוד של שלושה סוגי האוכל הנצרכים ביותר ע"י החיות.
10. הדפסת רשימת כל החיות עם המספר הרשומי שלהם שנולדו באותה שנה **printAimalsForGivenBirthYear** המקבלת שנה ומדפיסה את כל החיות שנולדו בשנת לידה זו.
11. מציאת מטפל **findEmployee** לפי אחד משני פרמטרים: ת.ז. או מספר החיות הנמצאות בטיפולו . אם יש יותר ממטפל אחד, יש למצוא את כולם ולבנות מהם רשימה מקושרת הממוינת לפי מספר ת.ז. *יש לבצע פעולה הזאת בצורה רקורסיבית.*
(שימו לב כי הכוונה כאן היא לכתוב שתי פונקציות שונות – האחת עבור ת.ז. והשניה עבור מספר החיות. כל אחת מהן רקורסיבית. ומעליהן להגדיר את הפונקציה findEmployee).

12. שאילתא **averageNumOfChildren** המחשבת את ממוצע הילדים של כל החיות בגן החיות.
הערה: יש לבצע פעולה זאת בצורה רקורסיבית. אין להשתמש בפונקציות עזר, משתנים גלובליים או סטטיים.

13. הדפסת רשימת כל המטפלים עם כל פרטיהם **printEmployee** המדפיסה את כל המטפלים עם כל פרטיהם.

14. מחיקת חיה מהמערכת **deleteAnimal** המקבלת מהמשתמש את המספר הרישומי של החיה ומוחקת אותה מהמערכת.
15. מחיקת **כל** חיות הגן מהמערכת **deleteAllAnimals**.
16.. מחיקת מטפל מרשימת המטפלים של החברה **deleteEmployee** המקבלת מהמשתמש את מספר הת.ז. ומוחקת אותו מהמערכת.
17. מחיקת רשימת כל המטפלים של גן החיות **deleteAllEmployees**.
18. מחיקת סוג האוכל מהמערכת **deleteFood** המקבלת מהמשתמש את הבר-קוד של האוכל ומוחקת אותו מהמערכת.
19. מחיקת **כל** סוגי האוכל מהמערכת **deleteAllFood**.

דגשים:

- בכל הפונקציות הרקורסיביות אתם מתבקשים לא להשתמש במשתנים סטטיים ופונקציות עזר (אלא אם כן כתוב במפורש שמותר לעשות זאת).
- אנא הקפידו להגדיר צומת בעץ שיכיל 3 שדות בלבד : מידע, מצביע לבן שמאלי ומצביע לבן ימני!

- **יש לבדוק את תקינות הקלט לכל הפונקציות. במקרה שהקלט לא תקין, יש להציג הודעה על שגיאה.**
- בדקו שהנכם מטפלים 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(קבצי C) וקבצי header (קבצי h) והן פונקציה ראשית main המדגימה את הבדיקות שנעשו לתוכנה שנכתבה. שימו לב כי פונקציה זאת צריכה להיות קצרה וקריאה!

הודעות שגיאה

סוגי השגיאות עליהן יש לדווח:
קלט לא תקין

הידור, קישור ובדיקה עצמית

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

1. 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:

עליכם להגיש קובץ מכווץ (zipped file), לפי הנחיות הבודק, שבו הקבצים המכילים את תוכניתכם (אותם אתם כתבתם) למודל.

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