החוג למדעי המחשב
המכללה האקדמית תל-חי

מבוא לתכנות מערכות
תרגיל בית מספר 7

נושאים: גרפים. פוינטרים לפונקציות.

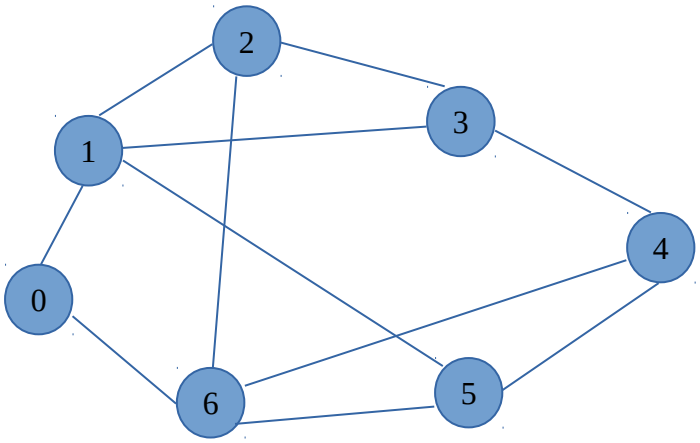
סמסטר אביב 2020-21

תאריך הגשה: 20.06.2021, שעה: 23:59
הגשה בזוגות

חלק 1 (גרפים)

קבוצת צמתים מתוך צמתי הגרף תקרא "קבוצה עיקרית", אם בכל צומת שלא נמצא בקבוצה יש קשת לצומת בתוך הקבוצה.

לדוגמא, עבור גרף G:



קבוצת הצמתים {0,3,5} היא "קבוצה עיקרית" כי הצמתים בגרף **שלא** בקבוצה הם {1,2,4,6}. עבור 1 יש קשת (1,0); עבור 2 יש קשת (2,3); עבור 4 יש קשת (4,5); עבור 6 יש קשת (6,0).

כתוב פונקציה המקבלת גרף G וקבוצת צמתים ומחזירה 1, אם הקבוצה הנתונה היא קבוצה עיקרית של G. אחרת, הפונקציה תחזיר 0.

int isMainGroup (int Graph[][], int graph_N, int Group[], int group_M)

כאשר

Graph[][] = מטריצת סמיכויות של G
graph_N = מספר הצמתים ב G
Group[] = קבוצת הצמתים שתבדק האם היא "קבוצה עיקרית"
group_M = מספר הצמתים בקבוצת הצמתים שתבדק האם היא "קבוצה עיקרית"

הנחות:

1. הגרף G הוא גרף לא מכוון ללא משקולות.
2. כל צמתי הגרף מחזיקים מספרים שלמים בים 0 ל N-1 (כאשר G בעל N צמתים).
3. הגרף G מתואר ע"י מטריצת סמיכויות.

חלק 2 (פוינטרים לפונקציות)

מטרת התרגיל

עבודה עם פוינטרים לפונקציות.
העבודה הנוכחית תהווה הרחבה של תרגיל בית 6 יחד עם השינויים הדרושים לשימוש **בעצים כלליים**.

תאור התרגיל

אתה מתבקש שוב לשפר תוכנה ישנה לניהול נתונים 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 כלל 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 6.
(**שים לב:** עליך לצמצם בצורה מרבית את הקוד שיצרת בתרגל בית מספר 6!)

אתם מתבקשים לשפר את המערכת באופן שיתואר להלן.

המערכת תכלול ניהול של 3 דברים: ניהול מלאי רכבים, ניהול ספקים, ניהול לקוחות.

1. כל **רכב** מאופיין ע"י 10 שדות:
   - מספר רישוי (מספר בן 7 ספרות)
   - מספר שלדה (מספר בן 5 ספרות)

- שם היצרן (שם יהיה מורכב ממילה אחת באורך ***לא ידוע***)
- שם הדגם (***באורך לא ידוע***)
- צבע (***באורך לא ידוע***)
- שנת יצור (מספר בן 4 ספרות)
- שנת עליה לכביש (מספר בן 4 ספרות)
- מחיר הרכב ששלום לספק (מספר בן עד 7 ספרות)
- המחיר הנוכחי של הרכב (מספר בן עד 7 ספרות)
- נפח מנוע (מספר בן 4 ספרות)

2. כל **ספק** מאופיין ע"י 5 שדות:
- מספר עוסק מורשה (מספר בן 10 ספרות)
- שם הספק (***באורך לא ידוע***)
- טלפון הספק (מספר בן 10 ספרות)
- מספר העסקאות עם הספק (מספר בן 5 ספרות)
- סכום כולל של עסקאות שנעשו עם הספק (מספר בן 10 ספרות)

3. כל **לקוח** מאופיין ע"י 7 שדות:
- שם פרטי (***באורך לא ידוע***)
- שם משפחה (***באורך לא ידוע***)
- ת.ז. (מספר שלם בן 9 ספרות)
- מספר הרישוי של הרכב שהושכר
- תאריך השכרת הרכב (בפורמט: dd/mm/yyyy)
- שעת השכרת הרכב (בפורמט: hh:mm)
- מחיר השכרת הרכב עבור 24 שעות בש"ח (מספר של 3 ספרות)

עליך לממש את הפעולות הבאות:

1. הגדרת עץ המכוניות **createCarTree** המגדירה ***עץ ריק*** של מכוניות.
רמז:
פונקציה זאת אמורה להכיל שורת קוד אחת – קריאה לפונקציה createTree עם פרמטרים מתאימים.

2. הוספת מכונית חדשה למערכת: **addNewCar** המקבלת מהמשתמש את כל הפרמטרים הדרושים ומגדירה מכונית חדשה במערכת.
רמז:
פונקציה זאת אמורה להכיל שורת קוד אחת – קריאה לפונקציה addNewNode עם פרמטרים מתאימים.

3. הגדרת עץ לקוחות החברה **createClientTree** המגדירה עץ ***ריק*** של הלקוחות באורך לא ידוע.
רמז:
פונקציה זאת אמורה להכיל שורת קוד אחת – קריאה לפונקציה createTree עם פרמטרים מתאימים.

4. הוספת לקוח חדש במערכת: **addNewClient** המקבלת מהמשתמש את כל הפרמטרים הדרושים של הלקוח ומגדירה לקוח חדש במערכת.
רמז:
פונקציה זאת אמורה להכיל שורת קוד אחת – קריאה לפונקציה addNewNode עם פרמטרים מתאימים.

5. הגדרת עץ ספקי <u>החברה</u> **createSupplierTree** המגדירה <u>עץ ***ריק***</u> של ספקים באורך לא ידוע.
רמז:
פונקציה זאת אמורה להכיל שורת קוד אחת – קריאה לפונקציה createTree עם פרמטרים מתאימים.

6. הוספת ספק חדש למערכת: **addNewSupplier** המקבלת מהמשתמש את כל הפרמטרים הדרושים של הספק ומגדירה ספק חדש במערכת.
רמז:
פונקציה זאת אמורה להכיל שורת קוד אחת – קריאה לפונקציה addNewNode עם פרמטרים מתאימים.

7. שאילתא של מספר הלקוחות של החברה המחזיקים ברכב משנת יצור נתונה **clientNumberWithGivenCarYear** המחזירה את מספר הלקוחות המחזיקים רכבים מאותה שנת יצור נתונה.

8. שאילתא של מספר הרכבים בעלי אותו נפח מנוע :
carNumberWithGivenCapacity המקבלת מספר המהווה נפח מנוע לבדיקה ומחזירה את מספר הרכבים בחברה בעלי הנפח הנ"ל.

9. מציאת לקוח **findClient** לפי אחד משני פרמטרים: ת.ז. או תאריך השכרת הרכב . אם יש יותר מלקוח אחד, יש למצוא את כולם ולבנות מהם רשימה מקושרת הממוינת לפי מספר ת.ז. *<u>יש לבצע פעולה הזאת בצורה רקורסיבית.</u>*

10. שאילתא **averageOfSupplierMoney** המחשבת את הסכום, בממוצע, של כל העסקאות שנעשו עם הספקים.
הערה: <u>יש לבצע פעולה זאת בצורה רקורסיבית. אין להשתמש בפונקציות עזר, משתנים גלובליים או סטטיים.</u>

11. שאילתא של 3 ספקים שאיתם נעשו עסקאות בהיקפים הגדולים ביותר:
threeGreatestSuppliers המחזירה את מספרי הרישוי של שלושה ספקים איתם נעשו העסקות כנ"ל.

12. הדפסת רשימת כל הלקוחות עם מספר הרכב שלהם
printClientCarsForGivenRentDate המקבלת תאריך ומדפיסה את כל הלקוחות ששכרו את רכבם בתאריך זה.

13. הדפסת רשימת כל הספקים עם כל פרטיהם
printSuppliers המדפיסה את כל הספקים עם כל פרטיהם.

14. מחיקת לקוח מהמערכת **deleteClient** המקבלת מהמשתמש את מספר ת.ז. של הלקוח ומוחקת אותו מהמערכת.

15. מחיקת <u>**כל**</u> לקוחות החברה מהמערכת **deleteAllClients**.

16. מחיקת רכב מרשימת הרכבים של החברה **deleteCar** המקבלת מהמשתמש את מספר הרישוי ומוחקת אותו מהמערכת.

17. מחיקת רשימת כל הרכבים של החברה **deleteAllCars**.

18. מחיקת ספק מהמערכת **deleteSupplier** המקבלת מהמשתמש את מספר עוסק המורשה של הספק ומוחקת אותו מהמערכת.

19. מחיקת **כל** ספקי החברה מהמערכת **deleteAllSuppliers**.

דגשים:

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!

יש לבצע קומפילציה בשרת המכללה ע"י הרצת הפקודה הבאה:

gcc mySource.c -ansi -Wall -pedantic-errors -lm -o myProg

כאשר mySource.c הוא קובץ או קבצי המקור, ו myProg הוא קובץ ההרצה הנוצר.

דרישות, הגבלות הערות רמזים ותוספות:

- יש להקפיד על תכנון נכון של התוכנית וכתיבה נכונה ב-C.

- יש לתעד את התוכנית.
- בשום אופן אין להשתמש במשתנים גלובאליים!

בפונקציות רקורסיביות:

- אין להשתמש במשתנים סטטיים
- אל תשכחו לבדוק תמיד (!) שהמצביע לעץ שקיבלתם אינו שווה ל NULL.
- למימוש הפונקציות הרקורסיביות עבדו עם שיטת ה"עוזרים". כלומר בקשו מ "עוזר שמאלי" לבצע את הפעולה הרקורסיבית ואחר כך מה"עוזר הימני" לבצע את הפעולה.

- שימו לב כי הרבה פונקציות הקשורות לעצים דומות זו לזו. ולכן מומלץ להתמקד קודם כל בעץ של פריט מסוים. לכתוב ולבדוק היטב את הפונקציות ורק אחר כך לעבור לכתיבה של עץ של פריט אחר. הדבר יחסוך לכם הרבה עבודה ותיקונים מיותרים.

הגשת התוכנית:

עליכם להגיש קובץ מכווץ (zipped file) שבו הקבצים המכילים את תוכניתכם (אותם אתם כתבתם) והן קובץ readme עם שמות שני המגישים למודל. יש להגיש את העבודה רק מחשבון של אחד השותפים.

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: Bad names of variables/functions (not meaningful, limited in length, not in English, etc.)
F: No or bad documentation (comments)/ the code is unreadable...
G: duplication of code.
H: Rare and generally serious errors

ערעורים:
יש להפנות לבודק **אביתר בהלקר** בלבד ע"י שליחת מייל **evyatarbhal@gmail.com** **תוך שבוע** ממועד פרסום הציונים. בכותרת המייל יש לציין:"ערעור במת"ם עבודת בית מס' X ע"י +ת.ז. של הסטודנטים". יש לקחת בחשבון שבעת הערעור העבודה נבדקת מחדש וכתוצאה מהערעור ציון העבודה עלול להשתנות (יכול לעלות או לרדת). התשובה לערעור תהיה סופית ולא ניתנת לערעור נוסף.