מבחן 10010 שאלון X

הוראות כלליות:

1. אין לשנות מבנים, הגדרות פונקציות ושורות קוד שכבר ניתנו
2. יש לממש כל שאלה בקבצים השייכים לה בלבד, אין להוסיף קבצים לפרויקט
3. אין להשתמש במשתנים גלובלים, בפונקציה exit וב goto
4. אין להגדיר מטריצות בצורה דינאמית ללא הקצאה דינאמית.
5. כל דבר שלא צוין שאסור אזי מותר

שאלה 1 (55 נקודות): (קבצים q1.h q1.c)

חברת טיולים צריכה להחליט לאיזה ארצות לפתוח טיולים. כל טיול הוא לארץ אחת בלבד. לצורך זה היא עושה סקר בין לקוחות אפשריים. מחיר הטיול משתנה בין מבוגר, ילד, תינוק או חייל ולכן כל לקוח צריך לציין איזה סוג לקוח הוא.

שדות הלקוח הם : שמו, סוגו ולאיזו ארץ הוא ישמח לנסוע.

שדות הטיול הם : ארץ הטיול, מספר הלקוחות המעוניינים בטיול לארץ זו ומערך מצביעים ללקוחות.

שדות חברת הטיולים : מערך מצביעים לטיולים ומספר הטיולים במערך

בקובץ q1.h נתונים המבנים, *enum* של סוג הלקוח וקבועים נוספים. **אין לשנות או להוסיף שדות למבנים אילו. בנוסף אין לשנות הגדרה של פונקציות שניתנו.**

ישנו קובץ בינארי בו רשומים לקוחות שענו לסקר, נתון ראשון הוא מספר הלקוחות שענו לסקר. ולאחר מכן כל הלקוחות. נתוני כל לקוח נשמרים בסדר הבא : שם לקוח, סוגו, שם הארץ בה ירצה לטייל .

מחרוזת נרשמת לקובץ בינארי באופן הבא :

- ערך ראשון הוא int המציין את מספר התווים במחרוזת כולל ה '\0'
- כל תווי המחרוזת כולל '\0'

החברה מייצרת מבנה בזיכרון המכיל את כל האינפורמציה הנמצאת בקובץ. היא לא יודעת מראש את מספר ושמות הארצות שיעלו בסקר. המערכים נמצאים תמיד במצב שבכל התאים יש נתונים כלומר אין להקצאות מראש גודל מסוים.

לאחר שהמערכת העלתה את כל הנתונים לזיכרון היא מעוניינת לחקור את הנתונים.

סעיף 1.1: (35 נקודות):

המערכת קוראת כל פעם לקוח אחד מהקובץ הבינארי ומוסיפה אותו למערך המצביעים לטיולים בחברת הטיולים באופן הבא :

אם היא מוצאת טיול שארץ הטיול שלו תואמת לארץ שהלקוח מעוניין היא מוסיפה אותו כלקוח נוסף לטיול זה. במידה ואין עדין טיול לארץ זו היא תפתח טיול חדש לארץ זו ותוסיף אותו כלקוח ראשון. כלומר במערך הטיולים יהיה טיול אחד לכל ארץ ובו כל הלקוחות שמעוניינים לטייל לארץ הזו.

נתונה הפונקציה הראשית

int fillCompanySurvey(Company pCompany);*

הקוראת את נתוני הלקוחות מהקובץ הבינארי לתוך המבנה Company. עליך להשלים את שתי הפונקציות הפנימיות :

סעיף 1.1.1 : **(10 נקודות)** – ממש את הפונקציה הבאה אשר קוראת לקוח אחד מהקובץ הבינארי

int readCustomerFromFile(FILE* fp,Customer* pCust);

הפונקציה מקבלת מצביע לקובץ הפתוח, fp, מצביע למבנה לקוח, pCust, ומחזירה 1 אם הצליחה ו 0 במקרה של כישלון.

סעיף 1.1.2 : **(25 נקודות)** – ממש את הפונקציה הבאה המוסיפה את הלקוח לחברה לפי ההסבר למעלה.

int addCustomerToCompany(Company* pCompany, Customer* pCust);

הפונקציה מקבלת מצביע למבנה חברה, pCompany, מצביע למבנה לקוח, pCust, ומחזירה 1 אם הצליחה ו 0 במקרה של כישלון.

סעיף 1.2: (8 נקודות):

נניח כי טיול עולה 1000 ₪. המערכת מחשבת מה יהיה סכום הגביה מכל טיול כשידוע כי מבוגר משלם 100% מהמחיר, ילד 50% , תינוק פטור מתשלום וחייל 80%. **נתונה הפונקציה** הראשית אשר מדפיסה את סכום הגביה מכל הטיולים :

void printTotalIncomePerContry(Company pCompany);*

ממש את הפונקציה הפנימית אשר מחשבת את סכום הגביה מטיול אחד. הפונקציה מקבלת מצביע לטיול , pTrip, ומחזירה את סכום הגביה הכולל.

float calculateIncome(Trip* pTrip);

סעיף 1.3: (12נקודות) :

ממש את הפונקציה **(Company* pComapny)** ***releaseCompany*** המקבלת מצביע למבנה חברה, pCompany, ומשחררת את כל ההקצאות בשאלה.

בקובץ הנתון הלקוחות הרשומים הם :

11 customers
David Adult France
Jon Child France
Helena Baby Germany
Lidia Adult Germany
Judith Soldier France
Zeus Adult, Italy
Josef Child Italy
Michel Baby France
Tal Adult Italy
Guy Soldier France
Moshe Child Germany

פלט סעיף ב' :

In country France the income is: 3100.00
In country Germany the income is: 1500.00
In country Italy the income is: 2500.00

שאלה 2 (45 נקודות): (קבצים q2.h q2.c)

נתונה מטריצת מספרים שלמים **M** בגודל **n** שורות ו-**m** עמודות. המטריצה נמצאת רציף בזיכרון, שורה אחרי שורה. בשאלה הזו ממיינים את השורות של המטריצה לפי קריטריונים שונים.

הערה: אין לשכפל קוד במימוש *שאלה מס' 2.1 ו 2.2*

רמזים לפתרונות אפשריים, (לא מחייבים):

1. אפשרות א': מגדירים מבנה עזר המכיל שדות – מספר שורה וערך להשוואה לפי הקריטריון המיון. מגדירים מערך של מבנים כאילו. ע"י מעבר על שורות המטריצה מאתחלים כל איבר במערך, ממיינים את מערך המבנים ואז בהתאמה מסדרים את שורות המטריצה. את המיון ניתן לבצע עם פונקציה **qsort.**
2. אפשרות ב': שימוש בפונקציות מיון ידועות תוך התאמה לבעיה המסוימת.

שאלה מס' 2.1: (15 נקודות)

ממש את הפונקציה ***sortMatrixRowsBySum*** הממיינת את השורות של מטריצה לפי סכום של כל האברים בשורה.

void sortMatrixRowsBySum (int *matrix, int rows, int columns)

הפונקציה מקבלת פרמטרים הבאים:

matrix - מצביע לאבר הראשון בשורה הראשונה;

rows, columns - מספר שורות ומספר עמודות במטריצה

שאלה מס' 2.2: (20 נקודות)

ממש את הפונקציה ***sortMatrixRowsByBits*** הממיינת את שורות המטריצה על פי כמות הסיביות לא אפסיות בתצוגה בינארית של כל האברים בשורה. לדוגמא: עבור שורה {1,3,7} מספר הסיביות בייצוג בינארי הוא 6.

void sortMatrixRowsByBits (int *matrix, int rows, int columns)

הפונקציה מקבלת פרמטרים הבאים:

matrix - מצביע לאבר הראשון בשורה הראשונה;

rows, columns - מספר שורות ומספר עמודות במטריצה

שאלה מס' 2.3: (10 נקודות)

ממש את הפונקציה **Q2():** הגדר המטריצה, מיין אותה תוך שימוש בפונקציות שפיתחת בסעיפים 2.1 ו-2.2. יש להדפיס את המטריצה לפני ואחרי כל מיון.