מבוא למדעי המחשב
מטלה 6
נושא: הקצאות דינמיות ומבנים
מתרגלים אחראים: ענבל אברהם ואביב שוקרון

הוראות כלליות

המטלה כוללת 3 קבצים:
mainTrain.c – קובץ ה-main שלכם, הוא מריץ בדיקות על הפונקציות שתממשו.
Application.h – קובץ הגדרות עבור המבנה Application והפונקציות הנחוצות לו.
AppStore.h – קובץ הגדרות עבור המבנה AppStore והפונקציות הנחוצות לו.

במטלה זו תתבקשו לממש מבנה נתונים (struct) ע"י מימוש של 12 פונקציות.
יש להוריד את המטלה ממערכת ההגשות האוטומטית ולהשתמש בקבצי ה-H המקוריים על מנת שלא יתגלו שגיאות קומפילציה מסיבות כמו שלא רשמתם שמות של פונקציות בצורה נכונה.

עליכם לייצר 2 קבצי C:
Application.c – בו תממשו את הפונקציות שיש ב-Application.h
AppStore.c – בו תממשו את הפונקציות שיש ב-AppStore.h

שימו לב!
בכל שאלה אתם רשאים להשתמש בקוד שמימשתם מסעיפים קודמים. למשל אם הפונקציה משאלה מספר 2 יכולה להשתמש בפונקציה שמימשתם בשאלה 1 אתם רשאים לקרוא לפונקציה מ-1 בפונקציה של 2.

לאחר שתממשו, תוכלו לקמפל ולהריץ את התכנית עם ה-main המוכן או לשנות את ה-main כראות עיניכם ולבצע עוד בדיקות משלכם.

חשוב!
יש לוודא הרצה של התכנית בסביבה שלכם לפני הגשה למערכת האוטומטית. בצורה זו תוכלו ללמוד ולהתפתח בצורה הטובה ביותר.

בסיום המטלה הנכם נדרשים להגיש **את 2 קבצי ה-C שיצרתם- Application.c ו-Appstore.c** ובהם המימושים של השאלות.

לא לשכוח לרשום הערות לאורך הקוד (כל 3-2 שורות).
כמו כן הנכם מתבקשים לרשום הערה ארוכה בתחילת הקובץ AppStore.c הכוללת שם, ת.ז ותאריך. כמו כן הנכם מתבקשים לרשום לפני כל פונקציה הערה ובה מה הפונקציה עושה.

ברצוננו לממש מאגר נתונים בסיסי עבור חנות אפליקציות.
לשם כך יש לנו את ההגדרות הבאות:

```
typedef struct Application
{
	char* name;		//the name of the application
	float cost;		//the cost of the application in dollars
	int downloads;		//how many users downloaded the app
}Application;

typedef struct AppStore
{
	Application**	apps;		//a pointer to an array of pointers to products
	int		num_of_apps;	//the number of different product types
	char*		name;		//the name of the store (apple, google, etc.)
}AppStore;
```

על מנת להחזיק את מאגר הנתונים הנ"ל ברצוננו לממש את הפונקציות הבאות.

1. Application* CreateNewApp(char* _name, float _cost, int _downloads);
הפונקציה מקבלת כפרמטר שם, עלות ומספר הורדות ומחזירה אפליקציה חדשה (שמוקצה דינמית) כאשר השדות של האפליקציה מאותחלים בהתאמה בפרמטרים שקיבלה מהפונקציה. לא לשכוח לבצע הקצאות לכל המשתנים שזקוקים לכך, על מנת למשל ששם האפליקציה יועתק ולא יצביע לאותו מקום בזיכרון כמו המקורי. במידה ואחת ההקצאות בפונקציה לא מצליחה יש לשחרר את הזיכרון שהוקצה ולא ניתן להשתמש בו (אם יש כזה) ולהחזיר NULL.

2. Application* DuplicateApp(Application* source);
הפונקציה מקבלת כפרמטר מצביע לאפליקציה ומחזירה אפליקציה חדשה (שמוקצה דינמית) בעלת נתונים זהים לאפליקציה שהתקבלה כפרמטר. לא לשכוח לבצע הקצאות לכל המשתנים שזקוקים לכך, על מנת למשל ששם האפליקציה יועתק ולא יצביע לאותו מקום בזיכרון כמו המקורי. במידה ואחת ההקצאות בפונקציה לא מצליחה יש לשחרר את הזיכרון שהוקצה ולא ניתן להשתמש בו (אם יש כזה) ולהחזירNULL .

3. AppStore* AddApp(AppStore* as, Application* app);
הפונקציה מקבלת מצביע לחנות ואפליקציה ומחזירה מצביע לחנות מעודכנת. במידה והמצביע as מצביע ל-NULL, יש ליצור חנות חדשה עם השם "Colman Store", עם אפליקציה אחת שתאותחל על ידי העתקה מתוכנה של האפליקציה app. לא לשכוח לבצע הקצאות לכל המשתנים שזקוקים לכך, על מנת למשל ששם האפליקציה יועתק ולא יצביע לאותו מקום בזיכרון כמו המקורי.
במידה ובחנות כבר יש אפליקציה עם שם זהה, הפונקציה רק תעדכן את כמות ההורדות של האפליקציה הקיימת בחנות לפי הכמות שהועברה ב-app.
במידה ובחנות אין אפליקציה עם שם כזה, נגדיל את מערך המצביעים של החנות, נוסיף אפליקציה חדשה ונעתיק את הנתונים מ-app לאפליקציה החדשה. במידה וההקצאה הצליחה, יש לעדכן את השדה num_of_apps.
במידה ואחת ההקצאות בפונקציה לא מצליחה יש לשחרר את הזיכרון שהוקצה ולא ניתן להשתמש בו (אם יש כזה) ולהחזירNULL .

4. AppStore* DuplicateStore(AppStore* source);
הפונקציה מקבלת מצביע לחנות קיימת source ועליה ליצור חנות חדשה מאותחלת בנתונים של source. יש לבצע הקצאה לכל המשתנים ולא העתקת מצביעים, כלומר להקצות לאפליקציות זיכרון חדש וגם לכל הנתונים הדורשים הקצאה. במידה ואחת ההקצאות בפונקציה לא מצליחה יש לשחרר את הזיכרון שהוקצה ולא ניתן להשתמש בו (אם יש כזה) ולהחזירNULL .

.5
void SortByCost(AppStore* as);
הפונקציה מקבלת מצביע לחנות as וממיינת את האפליקציות בחנות לפי מחיר האפליקציה מהקטן לגדול.

.6
void SortByName(AppStore* as);
הפונקציה מקבלת מצביע לחנות as וממיינת את האפליקציות בחנות לקסיקוגרפית לפי שם האפליקציה מהקטן לגדול.

.7
int TotalDownloads(AppStore* as);
הפונקציה מקבלת מצביע לחנות as ומחזירה את סך כל ההורדות מכל האפליקציות שיש בחנות (עוברת על כל האפליקציות בחנות וסוכמת את ההורדות שלהן).

.8
int UpdateDownloads(AppStore* as, char* name, int toAdd);
הפונקציה מקבלת מצביע לחנות as ושם name. הפונקציה מחפשת את האפליקציה עם השם הזה בחנות. במידה ומוצאת מוסיפה את toAdd לכמות ההורדות הקיימת ומחזירה 1. במידה והאפליקציה לא נמצאת, הפונקציה תחזיר 0.

.9
int UpdateCost(AppStore* as, char* name, float newCost);
הפונקציה מקבלת מצביע לחנות as ושם name. הפונקציה מחפשת את האפליקציה עם השם הזה בחנות. במידה ומוצאת מעדכנת את מחיר האפליקציה ל-newCost ומחזירה 1. במידה והאפליקציה לא נמצאת, הפונקציה תחזיר 0.

.10
int GetAppDownloads(AppStore* as, char* name);
הפונקציה מקבלת מצביע לחנות as ושם name. הפונקציה מחפשת את האפליקציה עם השם הזה בחנות. במידה ומוצאת מחזירה את כמות ההורדות של האפליקציה, במידה והאפליקציה לא נמצאת, הפונקציה תחזיר 1-.

.11
void FreeApp(Application* app);
הפונקציה מקבלת מצביע לאפליקציה ומשחררת את הזיכרון שהאפליקציה משתמשת בו.

.12
void FreeAppStore(AppStore* as);
הפונקציה מקבלת מצביע לחנות אפליקציות ומשחררת את כל נתוני החנות בצורה עמוקה (יש להשתמש בפונקציה מסעיף 11).