דו"ח פרויקט – קורס מבוא לתכנות מערכות
מערכת להשכרת רכבים

מגישים:
ליאל ביטון ת.ז 207000241
תומר קליינר ת.ז 207459009

1. <u>הקדמה:</u>

המערכת שפיתחנו במסגרת פרויקט זה מיועדת לחברות השכרת רכבים. המערכת מאפשרת ניהול של נכסי החברה, לרבות צי הרכבים, סניפי ההשכרה, לקוחות והשכרות פעילות. היא מספקת ממשק ידידותי לביצוע פעולות שונות **ע"י עובדי החברה** - כגון הוספת רכבים, לקוחות, השכרות וסניפים חדשים, עדכון פרטי השכרות קיימות ועוד.
המערכת שפיתחנו מיועדת עבור ניהול החברה ולכן תהיה בשימוש ע"י עובדי החברה ולא ע"י לקוחות פרטיים.
המערכת שלנו פועלת ב-3 מדינות בעולם : ישראל, תאילנד וארה"ב.
כמו כן, המערכת חדשה ולכן פועלת החל משנת 2024, מכאן ששנה זו היא המינימלית להזנת הזדמנות חדשות.

המערכת מורכבת ממבנים מרכזיים כגון כלי רכב, השכרה, סניף, חברת השכרה, לקוח, פוליסת ביטוח וחשבונית. היא מציעה מגוון פונקציות לניהול יעיל של פעילות החברה, ביניהן הדפסת רשימות של רכבים, סניפים, לקוחות והשכרות, הוספת פריטים חדשים, מיון וחיפוש של רכבים, מתן הנחות ללקוחות, חישוב הכנסות החברה ועוד.

2. <u>תיאור כל מבנה במערכת:</u>

שם המבנה	תיאור המבנה	הנתונים המוכלים במבנה
RentalCompany חברת השכרה <u>הישות הראשית</u>	מבנה זה אחראי להחזיק ולנהל את כל נכסי החברה – הכוללים צי רכבים, סניפים, לקוחות והשכרות. הפונקציות בקבצי המימוש של המבנה מאפשרות את ניהול כלל הפעולות בחברת ההשכרה.	שם החברה, רשימת סניפים, מערך כלי רכב, מערך לקוחות, מערך השכרות, מספר כלי רכב, מספר לקוחות, מספר השכרות, מספר סניפים ופרמטר מיון (מיועד למיון מערך כלי הרכב).
Vehicle כלי רכב	מבנה זה מתאר כלי רכב. המבנה הוא **פולימורפי** בהגדרתו בכך שהוא מכיל מספר תכונות שמאפיינות את כל כלי הרכב, בנוסף לתכונות המאפיינות סוגי כלי רכב ספציפיים, שהוגדרו **במבנים נפרדים**. לחברת השכרת הרכב יש 3 קטגוריות שונות של כלי רכב - Compact, Standard, Premium. כל קטגוריה מאופיינת בתכונות שייחודיות אך ורק לה. למשתמש יש את האפשרות להוסיף רכבים למערכת, כך שכל מופע של כלי רכב יהיה מאחת הקטגוריות בלבד.	מבנה ה"אב" של כלי הרכב מכיל מספר סידורי, מספר מושבים ברכב, סוג תיבת הילוכים, פרמטר לאינדיקציה האם הרכב תפוס, שנת ייצור, מד קילומטר, לוחית רישוי, עלות ליום, סוג קטגוריה, מצביע לפונקציית הדפסה ו- Union המשמש למימוש הפולימורפי. קטגוריית רכבי המיני מכילה יעילות דלק ויצרן. קטגוריית רכבי הסטנדרט מכילה תכולת תא מטען מקסימלית ויצרן. קטגוריית רכבי הפרימיום מכילה פרמטר לאינדיקציה האם הרכב חשמלי, שדרוג ויצרן.
Branch סניף	מבנה זה מתאר סניף של חברת ההשכרה.	סניף מכיל כתובת (מבנה בפני עצמו) ומס' מזהה סניף. המספרים המזהים לסניפים מאותחלים אוטומטית בסדר עולה לפי סדר הזנתם במערכת, החל מ-0.
Rental השכרה	מבנה זה אחראי לנהל השכרה בודדת – הכוללת מספר סידורי אוטומטי, כלי רכב, לקוח, ביטוח, חשבונית, תאריך התחלה וסיום, סכום העסקה, ומספר הסניף ממנו יאסוף הלקוח את הרכב הנבחר.	מספר סידורי אוטומטי, מצביע לכלי רכב, לקוח, תאריך תחילת ההשכרה, תאריך סיום ההשכרה, עלות משוקללת להזמנה, פוליסת ביטוח, חשבונית ומספר סידורי של הסניף ממנו ייאסף כלי הרכב המושכר.
Customer לקוח	מבנה זה מתאר לקוח בחברת ההשכרה.	תעודת זהות, שם פרטי, שם משפחה, מספר טלפון נייד וגיל הלקוח.
Invoice חשבונית	מבנה זה מתאר את החשבונית שמקבל הלקוח לאחר ביצוע הזמנה (השכרה).	מספר סידורי אוטומטי, מספר סידורי של ההשכרה עבורה יש להוציא חשבונית, עלות משוקללת ותאריך ביצוע ההזמנה.
Insurance פוליסת ביטוח	מבנה זה מתאר את תעודת הביטוח של הלקוח בהשכרה בודדת, כל השכרה חייבת בביטוח כלשהו. המערכת שלנו מאפשרת בחירה מבין 5 פוליסות ביטוח שונות : ביטוח בסיסי (חובה), ביטוח נזקים לרכב, ביטוח סיוע לצד הדרך, ביטוח תאונות אישיות וביטוח מקיף. כל פוליסה מכילה את ביטוח החובה. כל פוליסה מסופקת ללקוחות בעלות קבועה פר יום השכרה.	מספר סידורי אוטומטי, סוג פוליסת הביטוח ועלות הביטוח הנבחר ליום.

3. <u>שרטוט והסבר על אופן הדחיסה לקובץ הבינארי:</u>

בקובץ Insurance.h מוגדר המבנה Insurance המכיל מספר סידורי, סוג פוליסת ביטוח ועלות פוליסה ליום.
אנו נניח כי המערכת שלנו מסוגלת להכיל עד 10,000 השכרות פעילות.

- <u>משתנה InsuranceSN:</u> פוליסת ביטוח היא ייחודית עבור השכרה בודדת, השכרה לא יכולה לקבל שתי פוליסות שונות ופוליסה בודדת יכולה להיות מוכלת בהשכרה אחת בלבד, ולכן המספר הסידורי של פוליסת הביטוח יקבל ערכים בין 0-9999. **ניתן להשתמש ב-14 סיביות לשמירת משתנה זה.**

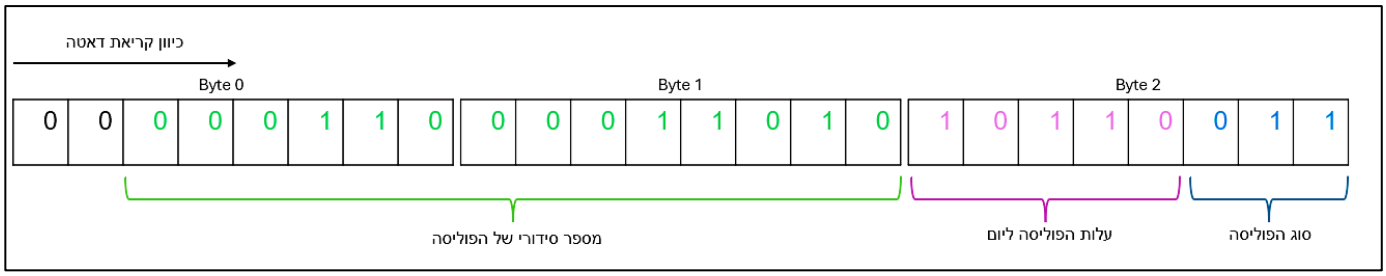
- <u>משתנה type:</u> סוג פוליסת הביטוח מיוצג באמצעות משתנה enum , במערכת שלנו ישנן 5 פוליסות ביטוח שונות מהן ניתן לבחור. ולכן משתנה זה ינוע בין הערכים 0-4. **ניתן להשתמש ב-3 סיביות לשמירת משתנה זה.**

- <u>משתנה costPerDay:</u> לכל סוג פוליסה יש עלות קבועה פר יום. העלות של הפוליסה היקרה ביותר היא 30 דולר ביום והעלות של הפוליסה הזולה ביותר היא 12 דולר ביום. ולכן משתנה זה ינוע בין הערכים 12-30.
ניתן להשתמש ב-5 סיביות לשמירת משתנה זה.

סה"כ 22 סיביות שיכולות להישמר ב-3 בתים.

נציג דוגמא: עבור פוליסת ביטוח מסוג "Personal Accident" שמספרה השלם הוא 3 (משתנה enum) נייצג באמצעות הסיביות **011**. מספרה הסידורי של הפוליסה הינו 1562 ונייצג אותו באמצעות הסיביות **11000011010**.
עלות פוליסה מהסוג הנבחר היא 22 דולר ונייצג אותה באמצעות הסיביות **10110**.

נשמור את סוג הפוליסה ב-3 הסיביות הנמוכות (ימניות) בבית השלישי וב-5 הסיביות הנותרות בבית זה נשמור את עלות סוג הפוליסה. בבית השני והראשון נשמור את המספר הסידורי של הפוליסה.



4. <u>הסבר על הפעולות אותן יכול המשתמש לבצע (פעולות התפריט):</u>
הפעולות מסודרות בסדר בו הן מופיעות בתפריט המערכת.

```
int initCompanyFromTFile(RentalCompany* pCompany, const char* fileName)
```

- <u>מטרה:</u> אתחול הדאטה במערכת מקובץ טקסט.
- <u>קלט:</u> הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany ואת שם הקובץ (קבוע).
- <u>פלט:</u> הפונקציה מחזירה 1 אם פעולת האתחול בוצעה בהצלחה. אחרת, מחזירה 0.

```
int initCompanyFromBFile(RentalCompany* pCompany, const char* fileName)
```

- <u>מטרה:</u> אתחול הדאטה במערכת מקובץ בינארי.
- <u>קלט:</u> הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany ואת שם הקובץ (קבוע).
- <u>פלט:</u> הפונקציה מחזירה 1 אם פעולת האתחול בוצעה בהצלחה. אחרת, מחזירה 0.

```
int initCompany(RentalCompany* company)
```

- <u>מטרה:</u> הפונקציה מאתחלת את חברת ההשכרה - קליטת שמה ואתחול כלל המערכים המכילים את רכיבי החברה.
- <u>קלט:</u> הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany וקולטת את שם החברה מהמשתמש.
- <u>פלט:</u> הפונקציה מחזירה 1 במידה והאתחול הצליח, אחרת מחזירה 0.

```
void printCompany(const RentalCompany* company)
```

- <u>מטרה:</u> הפונקציה מציגה את כלל נכסי חברת ההשכרה – צי הרכבים, לקוחות החברה, סניפי החברה והשכרות החברה.
- <u>קלט:</u> הפונקציה מקבלת מצביע למשתנה קבוע מסוג RentalCompany .
- <u>פלט:</u> הפונקציה תדפיס בצורה מסודרת וברורה את נכסי החברה.

void printAllBranches(const RentalCompany* company)
void printAllVehicles(const RentalCompany* company)
void printAllCustomers(const RentalCompany* company)
void printAllRentals(const RentalCompany* company)

- מטרה: פונקציות הדפסת המערכים ו/או רשימה שמטרתן להדפיס את כל ערכי המערך.
הפונקציות (למעט הפונקציה printAllBranches כיוון שהסניפים מיוצגים כרשימה) עושות שימוש בפונקציית מערך כללית.
- קלט: הפונקציות מקבלות מצביע למשתנה קבוע מסוג RentalCompany .
- פלט: הפונקציות מדפיסות את כל איבריהן תוך שימוש בפונקציות הדפסה מתאימות.

int addVehicle(RentalCompany* company)

- מטרה: הפונקציה מוסיפה רכב חדש למערך הרכבים של החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, הפונקציה קולטת מהמשתמש את הקטגוריה אליה משתייך הרכב אותו המשתמש רוצה להוסיף, ומאתחלת רכב חדש בהתאם לנתון זה. הפונקציה תעדכן את מספר הרכבים בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוסף רכב חדש בהצלחה, אחרת מחזירה 0.

int addCustomer(RentalCompany* company)

- מטרה: הפונקציה מוסיפה לקוח חדש למערך הלקוחות של החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, עושה שימוש בפעולה המאתחלת לקוח חדש, הפונקציה תעדכן את מספר הלקוחות בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוסף לקוח חדש בהצלחה, אחרת מחזירה 0.

int addRental(RentalCompany* company)

- מטרה: הפונקציה מוסיפה השכרה חדשה למערך ההשכרות של החברה. הפונקציה תאתחל השכרה חדשה בהתאם לתכונות הנדרשות.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, עושה שימוש בפעולה המאתחלת השכרה חדשה, הפונקציה תעדכן את מספר ההשכרות בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוספה השכרה חדשה בהצלחה, אחרת מחזירה 0.

int addBranch(RentalCompany* company)

- מטרה: הפונקציה מוסיפה סניף חדש לרשימת הסניפים של החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany ועושה שימוש בפעולה המאתחלת סניף חדש, הפונקציה תעדכן את מספר הסניפים בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוסף סניף חדש בהצלחה, אחרת מחזירה 0.

void sortVehicles(RentalCompany* company)

- מטרה: הפונקציה ממיינת את מערך הרכבים בחברה לבחירת המשתמש מתוך 4 תכונות על פיהן ניתן למיין – קילומטראז', מספר סידורי, שנת ייצור ולוחית רישוי.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany וקולטת מהמשתמש את פרמטר המיון בעזרת פעולת עזר.
- פלט: במידה ואין רכבים בחברה, הפונקציה מדפיסה זאת.

void findVehicle(const RentalCompany* company)

- מטרה: הפונקציה מחפשת (חיפוש בינארי) רכב לפי פרמטר המיון של מערך הרכבים.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany וקולטת מהמשתמש את הערך לחיפוש.
- פלט: במידה ונמצא הרכב, הפונקציה תדפיס את הרכב המתאים. אחרת, הפונקציה תדפיס שהרכב לא נמצא.

הפעולות היצירתיות שהוספנו:

void printRentalsByBranch(const RentalCompany* company)

- מטרה: הפונקציה תדפיס את כל ההשכרות שרלוונטיות לסניף מסוים.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג RentalCompany.
- פלט: הפונקציה תציג את ההשכרות שהסניף ממנו יאסוף הלקוח את הרכב הוא הסניף שנבחר.

void updateRental(RentalCompany* company)

- מטרה: הפונקציה מעדכנת השכרה קיימת בהתאם לפרמטר עדכון שנקלט מהמשתמש. הפונקציה מעדכנת את הפרמטרים הרלוונטיים בהשכרה לאחר השינוי. ניתן לעדכן תאריך תחילת השכרה, תאריך סיום השכרה, את הרכב המושכר וניתן לעדכן על סיום השכרה בתאריך של היום (הלקוח החזיר את הרכב לחברה "היום" – המערכת מסונכרנת עם שעון לוקלי).
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, קולטת מהמשתמש בעזרת פונקציית עזר את פרמטר העדכון.
- פלט: הפונקציה מדפיסה למסך הנחיות למשתמש ובדיקות תקינות נדרשות.

void RentalLotteryDiscount(RentalCompany* company)

- מטרה: הפונקציה מגרילה השכרה מתוך מערך ההשכרות ומזכה את הלקוח ב20% הנחה על הזמנתו.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany.
- פלט: במידה ואין השכרות להגריל מהן, הפונקציה תחזיר 0 ותודיע זאת למשתמש.
אחרת, הפונקציה תציג את הסכום לפני ההנחה שהתקבלה ואת ההזמנה (השכרה) שקיבלה הנחה ובה הסכום המעודכן.

void CalculateRevenueForSpecificYear(const RentalCompany* company)

- מטרה: הפונקציה מחשבת את מחזור ההכנסות השנתי של החברה, בשנה ספציפית שתיבחר ע"י המשתמש.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, הפונקציה קולטת מהמשתמש את השנה עבורה ירצה לראות את מחזור ההכנסות.
- פלט: הפונקציה מדפיסה את מחזור ההכנסות בשנה שנבחרה. במידה והשנה שנבחרה גדולה מהשנה הנוכחית (2024), הפונקציה תדפיס שלא ניתן לצפות במחזור הכנסות של שנה עתידית.

5. <u>שרטוט המערכת (מצורף בתיקיית ההגשה גם כקובץ **HTML** לנוחות הקריאה)</u>

השינויים שהוכנסו: הוספת כלל הפונקציות המעודכנות עבור כל מבנה, מימוש פולימורפיזם במבנה Vehicle והורדת מבנה Branches ובמקומו הכלת רשימת הסניפים במבנה הישות המרכזית RentalCompany.

6. <u>טבלת חלוקת תחומי אחריות:</u>

מבנה / תחום	מפתח
Rental Struct	ליאל ביטון
Customer Struct	ליאל ביטון
Invoice Struct	ליאל ביטון
Insurance Struct	ליאל ביטון
Binary Files	ליאל ביטון
Macros	יחד
Vehicle Struct	יחד
RentalCompany Struct	יחד
Branch Struct	תומר קליינר
Address Struct	תומר קליינר
Text Files	תומר קליינר
void* general Functions	תומר קליינר
Project Report & Diagram	יחד

על מנת לאפשר סקירה מלאה ומפורטת ככל הניתן על המערכת שפיתחנו, בהמשך הדו"ח ניתן לראות פירוט נרחב עבור כל מבנה במערכת ועבור כל הפונקציות אותן הוא מכיל.
כמו כן, בסוף הדו"ח ניתן לראות את פורמט הכתיבה לקובץ טקסט ו/או בינארי.

תודה רבה!

מבנה RentalCompany (חברת השכרה) – הישות הראשית של המערכת:

תיאור: מבנה זה אחראי להחזיק ולנהל את כל נכסי החברה – הכוללים צי רכבים, סניפים, לקוחות והשכרות. הפונקציות בקבצי המימוש של המבנה מאפשרות את ניהול כלל הפעולות בחברת ההשכרה.

תכונות:

- char* companyName;
- LIST branch_list;
- Vehicle** vehicleArr;
- Customer* customerArr;
- Rental* rentalArr;
- int numVehicles;
- int numCustomers;
- int numRentals;
- int numBranches;
- sortType sortBy;

פונקציות:

int initCompany(RentalCompany* company)

- מטרה: הפונקציה מאתחלת את חברת ההשכרה - קליטת שמה ואתחול כלל המערכים המכילים את רכיבי החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany וקולטת את שם החברה מהמשתמש.
- פלט: הפונקציה מחזירה 1 במידה והאתחול הצליח, אחרת מחזירה 0.

int addBranch(RentalCompany* company)

- מטרה: הפונקציה מוסיפה סניף חדש לרשימת הסניפים של החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany ועושה שימוש בפעולה המאתחלת סניף חדש, הפונקציה תעדכן את מספר הסניפים בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוסף סניף חדש בהצלחה, אחרת מחזירה 0.

int addVehicle(RentalCompany* company)

- מטרה: הפונקציה מוסיפה רכב חדש למערך הרכבים של החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, הפונקציה קולטת מהמשתמש את הקטגוריה אליה משתייך הרכב אותו המשתמש רוצה להוסיף, ומאתחלת רכב חדש בהתאם לנתון זה. הפונקציה תעדכן את מספר הרכבים בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוסף רכב חדש בהצלחה, אחרת מחזירה 0.

int addCustomer(RentalCompany* company)

- מטרה: הפונקציה מוסיפה לקוח חדש למערך הלקוחות של החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, עושה שימוש בפעולה המאתחלת לקוח חדש, הפונקציה תעדכן את מספר הלקוחות בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוסף לקוח חדש בהצלחה, אחרת מחזירה 0.

int addRental(RentalCompany* company)

- מטרה: הפונקציה מוסיפה השכרה חדשה למערך ההשכרות של החברה. הפונקציה תאתחל השכרה חדשה בהתאם לתכונות הנדרשות.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, עושה שימוש בפעולה המאתחלת השכרה חדשה, הפונקציה תעדכן את מספר ההשכרות בחברה.
- פלט: הפונקציה מחזירה 1 במידה ונוספה השכרה חדשה בהצלחה, אחרת מחזירה 0.

int chooseBranch(const RentalCompany* company)

- מטרה: הפונקציה מאפשרת בחירה של מספר סידורי של סניף מתוך רשימת הסניפים הקיימים, על מנת לקבוע את המספר הסידורי של הסניף ממנו ייאסף הרכב המושכר.
- קלט: הפונקציה מקבלת מצביע קבוע למשתנה מסוג RentalCompany, עושה שימוש בפעולה המחפשת סניף לפי מספר מזהה.
- פלט: הפונקציה תחזיר את המספר הסידורי של הסניף שנבחר, במידה ואין סניפים בחברה לבחור מהם הפונקציה תחזיר 1- .

Customer* chooseCustomerByID(const RentalCompany* company)

- מטרה: הפונקציה מאפשרת בחירה של לקוח מתוך מערך הלקוחות הקיימים, על מנת לקבוע את הלקוח המבצע השכרה.
- קלט: הפונקציה מקבלת מצביע קבוע למשתנה מסוג RentalCompany, עושה שימוש בפעולה המחפשת לקוח לפי תעודת זהות.
- פלט: הפונקציה תחזיר מצביע ללקוח שמצאה, במידה ואין לקוחות בחברה לבחור מהם הפונקציה תחזיר NULL.

Customer* findCustomerByID(Customer* customerArr, int numCustomers, char* ID)

- מטרה: הפונקציה מחפשת לקוח במערך הלקוחות, לפי תעודת זהות.
- קלט: הפונקציה מקבלת מערך של לקוחות, מספר לקוחות בחברה ואת תעודת הזהות שהזין המשתמש.
- פלט: במידה ומצאה לקוח עם תעודת הזהות שהזין המשתמש, הפונקציה מחזירה את המצביע אליו מסוג Customer , אחרת מחזירה NULL.

Branch* findBranchByID(const RentalCompany* company, int branchID)

- מטרה: הפונקציה מחפשת סניף ברשימת סניפים, לפי מספר מזהה.
- קלט: הפונקציה מקבלת מצביע קבוע למשתנה מסוג RentalCompany ואת המספר המזהה שהזין המשתמש.
- פלט: במידה ומצאה סניף עם המספר המזהה שהזין המשתמש, הפונקציה מחזירה את המצביע אליו מסוג Branch , אחרת מחזירה NULL.

void sortVehicles(RentalCompany* company)

- מטרה: הפונקציה ממיינת את מערך הרכבים בחברה לבחירת המשתמש מתוך 4 תכונות על פיהן ניתן למיין – קילומטראז', מספר סידורי, שנת ייצור ולוחית רישוי.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany וקולטת מהמשתמש את פרמטר המיון בעזרת פעולת עזר.
- פלט: במידה ואין רכבים בחברה, הפונקציה מדפיסה זאת.

void findVehicle(const RentalCompany* company)

- מטרה: הפונקציה מחפשת (חיפוש בינארי) רכב לפי פרמטר המיון של מערך הרכבים.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany וקולטת מהמשתמש את הערך לחיפוש.
- פלט: במידה ונמצא הרכב, הפונקציה תדפיס את הרכב המתאים. אחרת, הפונקציה תדפיס שהרכב לא נמצא.

sortType getSortType()

- מטרה: הפונקציה קולטת מהמשתמש את הערך על פיו ירצה למיין את מערך הרכבים.
- קלט: קליטת מספר ממשתמש.
- פלט: הפונקציה מחזירה את פרמטר המיון שבחר המשתמש, שהוא משתנה מסוג sortType.

void askUserSearchParameter(Vehicle* toSearch, const RentalCompany* company)

- מטרה: הפונקציה מעדכנת את הערך לחיפוש במשתנה זמני מסוג Vehicle.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Vehicle ומצביע קבוע למשתנה מסוג RentalCompany. הפונקציה קולטת מהמשתמש את הערך לחיפוש בהתאם לפרמטר המיון הנבחר.
- פלט: ללא.

int isVehicleAvailableInDates(const RentalCompany* company, const Vehicle* vehicle, const Date* start, const Date* end)

- מטרה: הפונקציה בודקת אם רכב מסוים זמין בתאריכים שנבחרו. הפונקציה עושה שימוש בפונקציית עזר הבודקת שתאריכים אינם מתנגשים.
- קלט: הפונקציה מקבלת משתנה קבוע מסוג RentalCompany, משתנה קבוע מסוג Vehicle ושני משתנים קבועים מסוג Date.
- פלט: הפונקציה מחזירה 1 אם הרכב הנבחר זמין בתאריכים שנבחרו. אחרת, מחזירה 0.

void updateRental(RentalCompany* company)

- מטרה: הפונקציה מעדכנת השכרה קיימת בהתאם לפרמטר עדכון שנקלט מהמשתמש. הפונקציה מעדכנת את הפרמטרים הרלוונטיים בהשכרה לאחר השינוי. ניתן לעדכן תאריך תחילת השכרה, תאריך סיום השכרה, את הרכב המושכר וניתן לעדכן על סיום השכרה בתאריך של היום (הלקוח החזיר את הרכב לחברה "היום" – המערכת מסונכרנת עם שעון לוקלי).
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, קולטת מהמשתמש בעזרת פונקציית עזר את פרמטר העדכון.

- פלט: הפונקציה מדפיסה למסך הנחיות למשתמש ובדיקות תקינות נדרשות.

int updateRentalHelper(RentalCompany* company)

- מטרה: הפונקציה מדפיסה את כלל ההשכרות בחברה הניתנות לעדכון תוך הצמדת מספר שורה לכל אחת לצורך בחירה נוחה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany.
- פלט: הפונקציה מחזירה 0 אם אין השכרות בחברה ומדפיסה זאת. אחרת, מחזירה 1.

int chooseIndexFromRentalArray(RentalCompany* company)

- מטרה: הפונקציה משמשת כפונקציית עזר לפעולת "עדכן השכרה". מטרתה לאפשר למשתמש לבחור את ההשכרה אותה ירצה לעדכן.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany וקולטת מהמשתמש מספר המשמש לאיתור ההשכרה אותה ירצה לעדכן.
- פלט: הפונקציה מחזירה את האינדקס המתאים לאיתור ההשכרה במערך.

updateType getUpdateType()

- מטרה: הפונקציה מאפשרת קליטת והחזרת פרמטר עדכון ההשכרה (תאריך התחלה/תאריך סיום/כלי רכב/סיום השכרה).
- קלט: הפונקציה קולטת מהמשתמש ערך מספרי המייצג את פרמטר העדכון הנבחר.
- פלט: הפונקציה מחזירה את פרמטר העדכון.

Customer* SelectNewOrExistingCustomer(RentalCompany* company)

- מטרה: בחירת הלקוח אשר מבצע השכרה חדשה. הפונקציה מאפשרת בחירה בין האפשרות של לקוח קיים במערכת לבין האפשרות של הכנסת לקוח חדש. הפונקציה שונה מהכנסת לקוח חדש, שכן היא משאירה את המשתמש בתהליך של הקמת השכרה חדשה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany , וקולטת מהמשתמש את בחירתו מבין שתי האופציות האפשריות.
- פלט: הפונקציה מחזירה מצביע מסוג Customer , ללקוח הנבחר שעבורו יש להקים השכרה חדשה במערכת.

void RentalLotteryDiscount(RentalCompany* company)

- מטרה: הפונקציה מגרילה השכרה מתוך מערך ההשכרות ומזכה את הלקוח ב20% הנחה על הזמנתו.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany.
- פלט: במידה ואין השכרות להגריל מהן, הפונקציה תחזיר 0 ותודיע זאת למשתמש.
אחרת, הפונקציה תציג את הסכום לפני ההנחה שהתקבלה ואת ההזמנה (השכרה) שקיבלה הנחה ובה הסכום המעודכן.

void CalculateRevenueForSpecificYear(const RentalCompany* company)

- מטרה: הפונקציה מחשבת את מחזור ההכנסות השנתי של החברה, בשנה ספציפית שתיבחר ע"י המשתמש.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany, הפונקציה קולטת מהמשתמש את השנה עבורה ירצה לראות את מחזור ההכנסות.
- פלט: הפונקציה מדפיסה את מחזור ההכנסות בשנה שנבחרה. במידה והשנה שנבחרה גדולה מהשנה הנוכחית (2024), הפונקציה תדפיס שלא ניתן לצפות במחזור הכנסות של שנה עתידית.

void printRentalsByBranch(const RentalCompany* company)

- מטרה: הפונקציה תדפיס את כל ההשכרות שרלוונטיות לסניף מסוים.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג RentalCompany.
- פלט: הפונקציה תציג את ההשכרות שהסניף ממנו יאסוף הלקוח את הרכב הוא הסניף שנבחר.

void printAllVehicles(const RentalCompany* company)
void printAllCustomers(const RentalCompany* company)
void printAllRentals(const RentalCompany* company)
void printAllBranches(const RentalCompany* company)

- מטרה: פונקציות הדפסת המערכים ו/או רשימה שמטרתן להדפיס את כל ערכי המערך.
הפונקציות (למעט הפונקציה printAllBranches כיוון שהסניפים מיוצגים כרשימה) עושות שימוש בפונקציית מערך כללית.
- קלט: הפונקציות מקבלות מצביע למשתנה קבוע מסוג RentalCompany .
- פלט: הפונקציות מדפיסות את כל איבריהן תוך שימוש בפונקציות הדפסה מתאימות.

void printCompany(const RentalCompany* company)

- מטרה: הפונקציה מציגה את כלל נכסי חברת ההשכרה – צי הרכבים, לקוחות החברה, סניפי החברה והשכרות החברה.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג RentalCompany .
- פלט: הפונקציה תדפיס בצורה מסודרת וברורה את נכסי החברה.

void freeVehicleArray(Vehicle** arr, int numVehicles)
void freeCustomerArray(Customer* arr, int numCustomers)

- מטרה: פונקציות שחרור המערכים שמטרתן לשחרר את המקום המוקצה בזיכרון לאיברי המערך בהתאם לפונקציית השחרור הרלוונטית.
- קלט: הפונקציות מקבלות את כתובת המערך ואת מספר הערכים בו.
- פלט: ללא.

void freeCompany(RentalCompany* company)

- מטרה: הפונקציה משחררת את המקום המוקצה בזיכרון למשתנים הרלוונטיים שלה הדרושים שחרור.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג RentalCompany.
- פלט: ללא.

מבנה Rental (השכרה):

תיאור: מבנה זה אחראי לנהל השכרה בודדת – הכוללת מספר סידורי אוטומטי, כלי רכב, לקוח, ביטוח, חשבונית, תאריך התחלה וסיום, סכום העסקה, ומספר הסניף ממנו יאסוף הלקוח את הרכב הנבחר.

תכונות:

- int rentalSN;
- Vehicle* vehicle;
- Customer customer;
- Date startDate;
- Date endDate;
- float totalCost;
- Insurance insurance;
- Invoice invoice;
- int branchID;

פונקציות:

int initRental(Rental* rental, Customer* customer, const Vehicle** vehicleArr, Rental* rentalArr, int numRentals, int numVehicles, int branchID)

- מטרה: הפונקציה מאתחלת השכרה חדשה.
- קלט: הפונקציה מקבלת מצביע למבנה Rental לאתחול, מצביע למבנה Customer, כתובת (מסוג קבוע) למערך כלי רכב, כתובת למערך השכרות, מספר השכרות בחברה, מספר כלי רכב בחברה ומספר זיהוי של סניף.
הפונקציה קולטת מהמשתמש את תאריכי ההשכרה ואת הרכב הרצוי.
הרכבים האפשריים המוצגים לבחירה הם הרכבים הפנויים בתאריכים שנבחרו.
- פלט: הפונקציה מחזירה 1 במידה ופעולת האתחול צלחה. אחרת, מחזירה 0.

Vehicle* findVehicleBySN(Vehicle** vehicleArr, int numVehicles, int SN)

- מטרה: הפונקציה מחפשת רכב במערך הרכבים, לפי מספר סידורי.
- קלט: הפונקציה מקבלת מערך של מצביעים לרכבים, מספר רכבים בחברה ומספר סידורי שהזין המשתמש.
- פלט: במידה ומצאה רכב עם המספר הסידורי שהזין המשתמש, הפונקציה מחזירה את המצביע אליו מסוג Vehicle , אחרת מחזירה NULL.

float calculateTotalCost(Rental* rental)

- מטרה: הפונקציה מחשבת את הסכום המשוקלל עבור השכרה בודדת, תוך התייחסות למספר ימי ההשכרה, עלות הרכב הנבחר פר יום ועלות הביטוח הנבחר פר יום.
- קלט: הפונקציה מקבלת מצביע למשתנה Rental .
- פלט: הפונקציה מחזירה מספר מסוג float המייצג את סך התשלום עבור ההשכרה.

int updateRentalGenerator(int num)

- מטרה: הפונקציה מעדכנת את המשתנה הסטטי המייצג מספר סידורי המאותחל באופן אוטומטי עבור השכרה בודדת.
- קלט: מספר שמתאר את ערך העדכון למשתנה הסטטי (לצורך עדכון המשתנה בעת טעינת דאטה).
- פלט: ערך המשתנה הסטטי + 1 להשמה הבאה.

int getCurrentRentalGenerator()

- מטרה: הפונקציה מאחזרת את ערך המשתנה הסטטי הנוכחי (לצורך בדיקה ועדכון משתנה בעת טעינת דאטה).
- קלט: ללא.
- פלט: ערך נוכחי של המשתנה הסטטי.

int isVehicleAvailableInDates(const Rental* rentalArr, int numRentals, const Vehicle* vehicle, const Date* start, const Date* end)

- מטרה: הפונקציה בודקת עבור רכב בודד, האם הוא זמין בתאריכים מסוימים.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג Rental ואת מספר ההשכרות בחברה, מקבלת מצביע למשתנה קבוע מסוג Vehicle ושני משתנים קבועים מסוג Date המייצגים תאריך התחלה וסיום של השכרה.
- פלט: הפונקציה מחזירה 0 אם הרכב שקיבלה אינו זמין להשכרה בתאריכים שהתקבלו. אחרת, מחזירה 1.

void printAvailableVehicles(const Vehicle** vehicleArr, const Rental* rentalArr, int numRentals, int numVehicles, const Rental* rental)

- מטרה: הפונקציה מציגה למשתמש את כל הרכבים הפנויים בתאריכים שהוזנו.
- קלט: הפונקציה מקבלת כתובת קבועה למערך כלי רכב, כתובת קבועה למערך ההשכרות בחברה, את מספר ההשכרות בחברה ואת מספר כלי הרכב בחברה. כמו כן, הפונקציה תקבל מצביע קבוע למשתנה מסוג Rental.
- פלט: הפונקציה מדפיסה את הרכבים הפנויים בתאריכי ההשכרה שנבחרו.

Vehicle* getVehicleInRental(const Vehicle** vehicleArr, Rental* rentalArr, int numRentals, int numVehicles, int rentalIndex)

- מטרה: הפונקציה מאפשרת את בחירת כלי הרכב הרצוי להשכרה.
- קלט: הפונקציה מקבלת כתובת קבועה למערך כלי הרכב, את מספר כלי הרכב בחברה, כתובת קבועה למערך ההשכרות בחברה ואת מספר ההשכרות בחברה. כמו כן, מקבלת את אינדקס ההשכרה במערך ההשכרות. הפונקציה קולטת מהמשתמש את המספר הסידורי של הרכב הרצוי.
- פלט: הפונקציה מציגה את כל הרכבים הפנויים בתאריכים של ההשכרה הממוקמת במקום האינדקס במערך. הפונקציה מחזירה מצביע לכלי הרכב הנבחר.

int endRental(Rental* rental)

- מטרה: הפונקציה "משחררת" את כלי הרכב שהושכר, ובנוסף מעדכנת את הקילומטראז של הרכב המושכר (ממוצע נסיעה של 100 ק"מ ליום).
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Rental.
- פלט: הפונקציה מציגה למשתמש את החשבונית שלו עבור ההשכרה.

void printRental(const Rental* rental)

- מטרה: הפונקציה מציגה פרטי השכרה בודדת.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג Rental.
- פלט: הפונקציה מדפיסה למסך את כלל פרטי ההשכרה – מספר סידורי של ההשכרה בחברה, הלקוח המשכיר, תאריכי ההשכרה, מספר הרכב המושכר ומספר הסניף ממנו יתבצע איסוף הרכב, פרטי הביטוח ופרטי החשבונית.

void freeRental(Rental* rental)

- מטרה: הפונקציה משחררת את המקום שהוקצה בזיכרון עבור המשתנים הרלוונטיים בהשכרה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Rental.
- פלט: ללא.

מבנה Address (כתובת):

תיאור: מבנה זה מתאר כתובת.
המערכת שפיתחנו פועלת ב-3 מדינות: ישראל, ארה"ב ותאילנד.

תכונות:

- eCountry country;
- char* city;
- char* street;
- int number;

פונקציות:

void initAddress(Address* address)

- מטרה: אתחול וקליטת כתובת מהמשתמש.
- קלט: מצביע למבנה Address , הפונקציה קולטת את הערכים הנדרשים מהמשתמש.
- פלט: הדפסות המנחות את המשתמש בהכנסת הערכים המתאימים.

void printAddress(const Address* address)

- מטרה: הדפסת כתובת.
- קלט: מצביע למבנה כתובת קבוע.
- פלט: הדפסת הכתובת בפורמט : [Street][number],[city],[country]

void freeAddress(Address* address)

- מטרה: שחרור המקום המוקצה בזיכרון למשתנים הרלוונטיים במבנה.
- קלט: מצביע למבנה Address.
- פלט: ללא.

מבנה Branch (סניף):

תיאור: מבנה המתאר סניף של חברת ההשכרה. לכל סניף יש כתובת, ומס' מזהה סניף. המספרים המזהים לסניפים מאותחלים אוטומטית בסדר עולה לפי סדר הזנתם במערכת, החל מ0.

תכונות:

- static int branchID_generator; (Global Variable)
- Address address;
- int branchID;

פונקציות:

int initBranch(Branch* branch)

- מטרה: לאתחל סניף.
- קלט: מצביע לסניף לאתחול. קלט מהמשתמש.
- פלט: הדפסות למשתמש להכוונת מילוי מרכיבי כתובת הסניף. מחזירה 1 אם הסניף אותחל בהצלחה. 0 אם לא הוקצה זכרון למבנה הנקלט.

void printBranch(const void* branch)

- מטרה: הדפסת סניף.
- קלט: מצביע גנרי קבוע.
- פלט: הדפסה של תכונות הסניף.

int updateBranchGenerator(int num)

- מטרה: עדכון המשתנה הסטטי, על מנת שהמספר המזהה של סניף יאותחל אוטומטית.
- קלט: מספר שמתאר את ערך העדכון למשתנה הסטטי.
- פלט: ערך המשתנה הסטטי בתוספת 1.

int getCurrentBranchGenerator()

- מטרה: אחזור ערך המשתנה הסטטי הנוכחי.
- קלט: ללא.
- פלט: ערך נוכחי של המשתנה הסטטי.

void freeBranch(Branch* branch)

- מטרה: שחרור סניף מהזכרון.
- קלט: מצביע למבנה סניף לשחרור.
- פלט: ללא.

מבנה Customer (לקוח):

תיאור: מבנה זה מתאר לקוח של החברה.

תכונות

- char ID[ID_LEN + 1];
- char* firstName;
- char* lastName;
- char phone[PHONE_LEN + 1];
- int age;

פונקציות:

int initCustomer(Customer* customer)

- מטרה: אתחול לקוח מהמשתמש.
- קלט: מצביע ללקוח לאתחול. קלט מהמשתמש.
- פלט: הדפסות להכוונת מילוי הנתונים של המשתמש. מחזירה 1 אם הלקוח אותחל בהצלחה. מחזירה 0 אחרת.

int getCustomerID(Customer* customer ,const Customer* customerArr, int numCustomers)

- מטרה: איתחול תעודת הזהות של לקוח חדש מהמשתמש ,תוך בדיקה שלקוח בעל תעודת זהות זהה לא קיים במערכת.
- קלט: מצביע ללקוח, מערך לקוחות החברה, מספר לקוחות בחברה. קלט מהמשתמש.
- פלט: הדפסות להכוונת מילוי הנתונים. מחזירה 1 ביציאה מהפונקציה.

int getPhoneNumber(Customer* customer)

- מטרה: איתחול מספר הטלפון של לקוח חדש מהמשתמש.
- קלט: מצביע ללקוח. קלט מהמשתמש.
- פלט: הדפסות להכוונת מילוי הנתונים. מחזירה 1 ביציאה מהפונקציה.

int getCustomerFullName(Customer* customer)

- מטרה: איתחול שם פרטי ושם משפחה של לקוח חדש מהמשתמש.
- קלט: מצביע ללקוח. קלט מהמשתמש.
- פלט: הדפסות להכוונת מילוי הנתונים. מחזירה 1 ביציאה מהפונקציה. מחזירה 0 אם אירעה שגיאה.

int checkUniqueID(const char* id, const Customer* customerArr, int numCustomers)

- מטרה: בדיקה שלקוח בעל תעודת זהות זהה לא קיים במערכת.
- קלט: מצביע לתעודת זהות קבועה לבדיקה , מערך לקוחות החברה, מספר לקוחות בחברה.
- פלט: מחזירה 0 אם קיים לקוח בעל תעודת זהות זהה במערך. מחזירה 1 אם תעודת הזהות ייחודית.

int getAge()

- מטרה: איתחול גיל של לקוח חדש מהמשתמש.
- קלט: קלט מהמשתמש..
- פלט: הדפסות להכוונת הזנת נתונים. מחזירה את הגיל.

int checkIfAllDigits(const char* Number, int len)

- מטרה: בדיקת תקינות קלט של מערך תווים שכולו ספרות.
- קלט: מערך תווים קבוע ואורכו.
- פלט: מחזירה 1 אם כל התווים הם ספרות. אחרת, מחזירה 0.

void printCustomer(const Customer* customer)

- מטרה: הדפסת לקוח.
- קלט: מצביע ללקוח קבוע.
- פלט: הדפסה של פרטי הלקוח.

void freeCustomer(Customer* customer)

- מטרה: שחרור המשתנים הרלוונטיים של הלקוח מהזיכרון.
- קלט: מצביע ללקוח לשחרור.
- פלט: ללא.

מבנה Vehicle (כלי רכב):

תיאור: מבנה זה מתאר כלי רכב. המבנה הוא **פולימורפי** בהגדרתו בכך שהוא מכיל מספר תכונות שמאפיינות את כל כלי הרכב, בנוסף לתכונות המאפיינות סוגי כלי רכב ספציפיים, שהוגדרו במבנים נפרדים.

לחברת השכרת הרכב יש 3 קטגוריות שונות של כלי רכב - Compact, Standard, Premium.
כל קטגוריה מאופיינת בתכונות שייחודיות אך ורק לה. למשתמש יש את האפשרות להוסיף רכבים למערכת, כך שכל מופע של כלי רכב יהיה מאחת הקטגוריות בלבד.
להלן פירוט הקטגוריות:

רכבי Compact:
רכבים אלו קטנים וחסכוניים בדלק, ולכן תכונותיהם:

- double fuelEfficiency;
- eCompactBrand brand;

חברת ההשכרה מקבלת רכבי Compact מ4 יצרניות שונות:
Volkswagen, Kia, Nissan, Hyundai

רכבי Standard:
רכבים אלו הם רכבים משפחתיים עם תא מטען רחב, ולכן תכונותיהם:

- double cargoCapacity;
- eStandardBrand brand;

חברת ההשכרה מקבלת רכבי Standard מ3 יצרניות שונות:
Toyota, Honda, Ford

רכבי Premium:
רכבים מסוג זה יוקרתיים, לעיתים חשמליים, ומאופיינים בשדרוגים מיוחדים שאין לשאר הרכבים, לכן תכונותיהם :

- int isElectric;
- eFeatures feature;
- ePremiumBrand brand;

חברת ההשכרה מקבלת רכבי Premium מ3 יצרניות שונות:
Mercedes-Benz, Audi, Lexus

רכבי Premium מגיעים עם אחד מהשדרוגים הבאים:
ריפוד יוקרתי, טכנולוגיה מתקדמת, בטיחות מוגברת, כל השדרוגים יחד או ללא שדרוג.

תכונות המבנה Vehicle:

- void (*print)(const struct Vehicle_*);
- int vehicleSN;
- int numSeats;
- int gearBox;
- int isTaken;
- int year;
- double odometer;
- char licensePlate[MAX_PLATE + 1];
- int costPerDay;
- eCategory categoryType;

// Union to hold different categories

- union {
  Premium premium;
  Standard standard;
  Compact compact;
  } category;

המבנה מחזיק לכל כלי רכב את המספר הסידורי שלו במערכת, כמות המושבים, תיבת הילוכים (בוליאני (0 או 1)-ידני או אוטומטי), האם הרכב תפוס (בוליאני – 0 או 1), שנת ייצור, קילומטראז', לוחית רישוי, ומחיר יומי.
בנוסף המבנה מחזיק לכל כלי רכב את הקטגוריה שלו, את פונקציית ההדפסה שלו ואת כל הקטגוריות האפשריות תחת union.
בעת אתחול רכב, פונקציית ההדפסה, הקטגוריה והתכונות הקטגוריאליות מוגדרות בהתאם לקטגוריה שבחר המשתמש.

פונקציות:

void initPremium(Vehicle** vehicleArr, Vehicle* premium, int vehicleCount)

- מטרה: אתחול תכונות רגילות לרכב בנוסף לתכונות של רכב פרימיום.
- קלט: מערך מצביעים לרכבים, מצביע לרכב, מספר הרכבים בחברה. קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים.

void initStandard(Vehicle** vehicleArr, Vehicle* standard, int vehicleCount)

- מטרה: אתחול תכונות רגילות לרכב בנוסף לתכונות של רכב סטנדרט.
- קלט: מערך מצביעים לרכבים, מצביע לרכב, מספר הרכבים בחברה. קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים.

void initCompact(Vehicle** vehicleArr, Vehicle* compact, int vehicleCount)

- מטרה: אתחול תכונות רגילות לרכב בנוסף לתכונות של רכב קומפקט.
- קלט: מערך מצביעים לרכבים, מצביע לרכב, מספר הרכבים בחברה. קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים.

void initVehicle(Vehicle** vehicleArr, Vehicle* vehicle, int vehicleCount)

- מטרה: אתחול תכונות רגילות לרכב
- קלט: מערך מצביעים לרכבים, מצביע לרכב, מספר הרכבים בחברה. קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים.

void printVehicle(const Vehicle* vehicle)

- מטרה: הדפסת תכונות רגילות של רכב.
- קלט: מצביע קבוע לרכב.
- פלט : הדפסת כלי הרכב.

void printPremium(const Vehicle* premium)

- מטרה: הדפסת תכונות רגילות של רכב בנוסף לתכונות של רכב פרימיום.
- קלט: מצביע קבוע לרכב.
- פלט: הדפסת פרטי הרכב.

void printStandard(const Vehicle* standard)

- מטרה: הדפסת תכונות רגילות של רכב בנוסף לתכונות של רכב סטנדרט
- קלט: מצביע קבוע לרכב.
- פלט: הדפסת פרטי הרכב.

void printCompact(const Vehicle* compact)

- מטרה: הדפסת תכונות רגילות של רכב בנוסף לתכונות של רכב קומפקט
- קלט: מצביע קבוע לרכב.
- פלט: הדפסת פרטי הרכב.

void printVehicleV(void* vehicle)

- מטרה: לאפשר שימוש בפונקציות גנריות והדפסת כלי רכב לפי פונקציית ההדפסה של הקטגוריה שלו.
- קלט: מצביע גנרי.
- פלט: הדפסת פרטי הרכב.

void getLicensePlate(Vehicle* vehicle)

- מטרה: קליטת לוחית הרישוי לרכב מהמשתמש
- קלט: מצביע לרכב. קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים.

int getNumSeats()

- מטרה: קליטת מס' מושבים ברכב מהמשתמש.
- קלט: קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים.

int getvehicleYear()

- מטרה: קליטת שנת ייצור של הרכב מהמשתמש.
- קלט: קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים.

int getVehicleSN(Vehicle** vehicleArr, int vehicleCount)

- מטרה: קליטת מספר סידורי לרכב מהמשתמש.
- קלט: מערך מצביעים לרכבים, מספר הרכבים בחברה. קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים. מחזירה את המספר הסידורי של הרכב.

int getVehicleBrand(char** arrName, int numOfOpt)

- מטרה: קליטת מספר המייצג את יצרן הרכב.
- קלט: מערך מחרוזות, מספר הקטגוריות הקיימות. קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים. מחזירה את האפשרות שנבחרה.

eFeatures getPremiumFeature()

- מטרה: קליטת סוג השדרוג לרכב פרימיום.
- קלט: קלט מהמשתמש.
- פלט: הדפסות להכוונת הזנת הנתונים. מחזירה את האפשרות שנבחרה.

void updateOdometer(Vehicle* vehicle, int totalDays, int kmPerDay)

- מטרה: עדכון הקילומטראז' ברכב, נעשה שימוש בפונקציה זו בעת סיום השכרה.
- קלט: מצביע לרכב, ימי השכרה, קילומטרים ביום.
- פלט: ללא.

int rentVehicle(Vehicle* vehicle)

- מטרה: לעדכן את ערך isTaken של רכב.
- קלט: מצביע לרכב
- פלט: מחזירה 1.

int checkUniqueSN(int SN, Vehicle** vehicleArr, int vehicleCount)

- מטרה: בדיקת ייחודיות של מספר סידורי במערך רכבים של החברה.
- קלט: מערך מצביעים לרכבים, מספר הרכבים בחברה, מספר סידורי לבדיקה
- פלט: מחזירה 1 אם המספר סידורי לא קיים כבר במערכת. מחזירה 0 אחרת.

int checkUniquePlate(char* plate, Vehicle** vehicleArr, int vehicleCount)

- מטרה: בדיקת ייחודיות של לוחית רישוי במערך רכבים של החברה.
- קלט: מערך מצביעים לרכבים, מספר הרכבים בחברה , מערך תווים של לוחית רישוי
- פלט: מחזירה 1 אם המספר סידורי לא קיים כבר במערכת. מחזירה 0 אחרת.

int compareByOdometer(const void* v1, const void* v2)

- מטרה: לאפשר השוואה בין 2 רכבים על בסיס קילומטראז'.
- קלט: 2 מצביעים גנריים
- פלט: 0 – קילומטראז' של v1 שווה לקילומטראז' של v2
  0 > – קילומטראז' של v1 קטן מהקילומטראז' של v2
  0< – קילומטראז' של v1 גדול מהקילומטראז' של v2

int compareBySN(const void* v1, const void* v2)

- מטרה: לאפשר השוואה בין 2 רכבים על בסיס מספר סידורי.
- קלט: 2 מצביעים גנריים
- פלט: 0 – מספר סידורי של v1 שווה למספר סידורי של v2
  0 > – מספר סידורי של v1 קטן מהמספר סידורי של v2
  0< – מספר סידורי של v1 גדול מהמספר סידורי של v2

int compareByYear(const void* v1, const void* v2)

- מטרה: לאפשר השוואה בין 2 רכבים על בסיס שנת ייצור.
- קלט: 2 מצביעים גנריים
- פלט: 0 – שנת ייצור של v1 שווה לשנת ייצור של v2
  0 > – שנת ייצור של v1 קטנה משנת ייצור של v2
  0< – שנת ייצור של v1 גדולה משנת ייצור של v2

int compareByLicensePlate(const void* v1, const void* v2)

- מטרה: לאפשר השוואה בין 2 רכבים על בסיס לוחית רישוי. ההשוואה מתבצעת לפי ערך ASCII
- קלט: 2 מצביעים גנריים
- פלט: 0 – לוחית רישוי של v1 שווה ללוחית רישוי של v2
  0 > – לוחית רישוי של v1 קטנה מלוחית רישוי של v2
  0< – קילומטראז של v1 גדולה מלוחית רישוי של v2

void freeVehiclePtr(void* pVehiclePtr)

- מטרה: לאפשר שחרור של כלי רכב מהזיכרון בעזרת פונקציית מערך גנרית.
- קלט: מצביע גנרי.
- פלט: ללא.

מבנה Invoice (חשבונית):

תיאור: מבנה זה מתאר את החשבונית שמקבל הלקוח בסוף ההשכרה.

תכונות:

- int invoiceSN;
- int rentalSN;
- float totalAmount;
- Date issueDate;

פונקציות:

int createInvoice(Invoice* invoice ,float amount, int rentalSerial)

- מטרה: הפונקציה מאתחלת חשבונית חדשה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Invoice את סכום ההשכרה הכולל ואת המספר הסידורי של ההשכרה שעבורה יש צורך להוציא חשבונית. הפונקציה קולטת מהמשתמש את תאריך ביצוע ההזמנה.
- פלט: הפונקציה מחזירה 1 אם פעולת יצירת החשבונית בוצעה בהצלחה. אחרת, מחזירה 0

int updateInvoiceGenerator(int num)

- מטרה: הפונקציה מעדכנת את המשתנה הסטטי המייצג מספר סידורי המאותחל באופן אוטומטי עבור חשבונית בודדת.
- קלט: מספר שמתאר את ערך העדכון למשתנה הסטטי (לצורך עדכון המשתנה בעת טעינת דאטה).
- פלט: ערך המשתנה הסטטי + 1 להשמה הבאה.

int getCurrentInvoiceGenerator()

- מטרה: הפונקציה מאחזרת את ערך המשתנה הסטטי הנוכחי (לצורך בדיקה ועדכון משתנה בעת טעינת דאטה).
- קלט: ללא.
- פלט: ערך נוכחי של המשתנה הסטטי.

void printInvoice(const Invoice* invoice)

- מטרה: הפונקציה מציגה את פרטי החשבונית.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג Invoice.
- פלט: הפונקציה מדפיסה למסך את פרטי החשבונית – המספר הסידורי של החשבונית + המספר הסידורי של ההשכרה אליה שייכת החשבונית, הסכום הכולל לתשלום ותאריך ביצוע העסקה.

void updateInvoice(Invoice* invoice, float updateCost)

- מטרה: הפונקציה תעדכן את הסכום לתשלום בחשבונית (במידה והתעדכנו פרטי ההשכרה או במידה והלקוח קיבל הנחה).
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Invoice וסכום מעודכן לתשלום.
- פלט: ללא.

מבנה Insurance (פוליסת ביטוח):

תיאור: מבנה זה מתאר את תעודת הביטוח של הלקוח בהשכרה בודדת, כל השכרה חייבת בביטוח כלשהו.
המערכת שלנו מאפשרת בחירה מבין 5 פוליסות ביטוח שונות : ביטוח בסיסי (חובה), ביטוח נזקים לרכב, ביטוח סיוע לצד הדרך, ביטוח תאונות אישיות וביטוח מקיף. כל פוליסה מכילה את ביטוח החובה.
כל פוליסה מסופקת ללקוחות בעלות קבועה פר יום השכרה:
ביטוח חובה – 12 דולר.
ביטוח נזקים לרכב – 20 דולר.
ביטוח סיוע לצד הדרך – 18 דולר.
ביטוח תאונות אישיות – 22 דולר.
ביטוח מקיף – 30 דולר.

תכונות:

- unsigned int InsuranceSN;
- eType type;
- unsigned int costPerDay;

פונקציות:

int createInsurance(Insurance* insurance)

- מטרה: הפונקציה מאתחלת פוליסת ביטוח חדשה ללקוח בהשכרה.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Insurance וקולטת מהמשתמש את פוליסת הביטוח הרצויה.
- פלט: הפונקציה מחזירה 1 אם פעולת יצירת הביטוח הצליחה. אחרת, מחזירה 0.

eType getInsuranceType()

- מטרה: הפונקציה מציגה למשתמש את אפשרויות הביטוח, קולטת ממנו את בחירתו ומחזירה אותה.
- קלט: הפונקציה קולטת מהמשתמש את סוג פוליסת הביטוח הרצויה.
- פלט: הפונקציה מחזירה משתנה מסוג eType שהינו enum המייצג את הפוליסה שנבחרה.

unsigned int updateInsuranceGenerator(unsigned int num)

- מטרה: הפונקציה מעדכנת את המשתנה הסטטי המייצג מספר סידורי המאותחל באופן אוטומטי עבור פוליסת ביטוח בודדת.
- קלט: מספר שמתאר את ערך העדכון למשתנה הסטטי (לצורך עדכון המשתנה בעת טעינת דאטה).
- פלט: ערך המשתנה הסטטי + 1 להשמה הבאה.

unsigned int getCurrentInsuranceGenerator()

- מטרה: הפונקציה מאחזרת את ערך המשתנה הסטטי הנוכחי (לצורך בדיקה ועדכון משתנה בעת טעינת דאטה).
- קלט: ללא.
- פלט: ערך נוכחי של המשתנה הסטטי.

const char* GetTypeStr(int type)

- מטרה: הפונקציה מאפשרת אחזור טקסטואלי של סוג פוליסת הביטוח המבוקשת.
- קלט: הפונקציה מקבלת מספר שלם המייצג את פוליסת הביטוח הרצויה.
- פלט: המחזירה מחזירה משתנה קבוע מסוג char*.

void printInsurance(const Insurance* insurance)

- מטרה: הפונקציה מציגה את פרטי פוליסת הביטוח.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג Insurance.
- פלט: הפונקציה מדפיסה למסך את פרטי הפוליסה – המספר הסידורי של הפוליסה ואת סוג הפוליסה.

מבנה Date (תאריך):

תיאור: מבנה זה מתאר תאריך (יום, חודש, שנה) ומשמש במערכת שלנו עבור תאריכי תחילת וסוף השכרת הרכב, ועבור תאריך ביצוע הזמנה. במערכת שלנו המבנה משמש לתאריכים עתידיים, כיוון שלא ניתן לבצע הזמנה לתאריכים שעברו, ולכן השנה המינימלית האפשרית היא השנה הנוכחית – 2024. אנו יוצאים מנקודת הנחה שמערכת זו חדשה בחברת ההשכרה החל משנה זו וכלל התאריכים שיוזנו יהיו עבור הזמנות חדשות בשנת 2024.

תכונות:

- int day;
- int month;
- int year;

פונקציות:

int getCorrectDate(Date* date)

- מטרה: אתחול והזנת תאריך חדש בפורמט מתאים.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Date. הפונקציה קולטת מהמשתמש את התאריך הרצוי.
- פלט: ללא.

int checkDate(char* date, Date* pDate)

- מטרה: הפונקציה בודקת שהתאריך שהוזן עומד בתנאי התקינות של תאריך בכלל ושל הפורמט המבוקש, ובמידה וכן, מאתחלת את ערכי התאריך לערכים שהוזנו.
- קלט: הפונקציה מקבלת משתנה מסוג *char המייצג את הערכים שהזין המשתמש. בנוסף, מקבלת מצביע למשתנה מסוג Date אליו תאתחל הפונקציה את הערכים התקינים.
- פלט: במידה והערכים שהזין המשתמש אינם תקינים (פורמט ו/או ערכי התאריך) הפונקציה מחזירה 0. אחרת, תחזיר 1.

int calculateDaysOfRental(const Date start, const Date end)

- מטרה: הפונקציה מחשבת את מספר הימים בין שני תאריכים (התחלה וסיום השכרה).
- קלט: הפונקציה מקבלת שני משתנים מסוג Date המייצגים תאריך התחלה ותאריך סיום.
- פלט: במידה ומספר הימים הוא 0 (כלומר, ההשכרה היא ליום אחד בודד, בוקר עד ערב) הפונקציה תחזיר 1. אחרת, הפונקציה תחזיר את מספר הימים שחושבו.

int dateRangesDoNotCollide(const Date* start1, const Date* end1, const Date* start2, const Date* end2)

- מטרה: הפונקציה בודקת עבור שני טווחי תאריכים האם קיימת התנגשות/הכלה/חפיפה ביניהם. הפונקציה משמשת כלי עזר לבדיקת והדפסת הרכבים הפנויים בחברה.
- קלט: הפונקציה מקבלת 4 מצביעים למשתנים קבועים מסוג Date כאשר כל זוג משתנים מייצג טווח תאריכים אחר (התחלה וסיום).
- פלט: הפונקציה מחזירה 1 במידה ואין התנגשות/חפיפה בטווחי התאריכים. אחרת, מחזירה 0.

int checkRentDates(const Date start, const Date end)

- מטרה: הפונקציה בודקת שטווח התאריכים הוא טווח תקין (שתאריך סיום ההשכרה אינו מוקדם יותר מאתריך התחלת ההשכרה).
- קלט: הפונקציה מקבלת שני משתנים קבועים מסוג Date.
- פלט: הפונקציה מחזירה 0 במידה ותאריך הסיום מוקדם יותר מתאריך ההתחלה. אחרת, מחזירה 1.

int getTodaysDate(Date* date)

- מטרה: הפונקציה מחזירה את התאריך של היום.
- קלט: הפונקציה מקבלת מצביע למשתנה מסוג Date ומעדכנת את ערכיו לתאריך העדכני של היום.
- פלט: הפונקציה מחזירה 1 אם תהליך העדכון בוצע בהצלחה.

int isDatePassedOrToday(const Date* date)

- מטרה: הפונקציה בודקת האם התאריך שהתקבל הוא התאריך של היום או תאריך של עבר.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג Date.
- פלט: הפונקציה מחזירה 1 אם התאריך שהתקבל הוא התאריך של היום או תאריך שעבר. אחרת, מחזירה 0.

```
void printDate(const Date* date)
```

- מטרה: הדפסת התאריך.
- קלט: הפונקציה מקבלת מצביע למשתנה קבוע מסוג Date.
- פלט: הפונקציה מדפיסה למסך את התאריך בפורמט הבא: dd/mm/yyyy

פורמט כתיבה לקובץ טקסט:

*פורמט הכתיבה לקובץ בינארי זהה לפורמט זה למעט שינויים מינוריים שהם:

- המבנה "פוליסת ביטוח" נכתב לקובץ בצורה דחוסה המפורטת מעלה.
- המבנים "תאריך" ו-"חשבונית" נכתבו ישירות לקובץ הבינארי ללא הפרדת שדות. באותו אופן, כך נכתבו גם הקטגוריות השונות של כלי הרכב.
- לפני כל כתיבה של משתנה מחרוזת מסוג *char נכתבה כמות התווים במחרוזת לקובץ.

פורמט כתיבה **"חברת השכרה"** לקובץ:
[שם חברת ההשכרה] – מחרוזת
[מערך כלי רכב]
[מערך לקוחות]
[מערך השכרות]
[רשימת סניפים]

פורמט כתיבה של **מערך כלי רכב** לקובץ:
[כמות כלי הרכב] – מספר שלם
[**כלי רכב** 1]
[**כלי רכב** 2]
...
[**כלי רכב** n]

כאשר כל **כלי רכב** ייכתב כך:
[מספר סידורי של כלי הרכב בחברה] – מספר שלם
[מספר המושבים] – מספר שלם
[תיבת הילוכים] – מספר 0/1
[האם תפוס] – מספר 0/1
[שנת ייצור] – מספר שלם
[קילומטראז] – מספר עשרוני (double)
[עלות ליום] – מספר שלם
[סוג קטגוריית הרכב] – מספר שלם (enum)
[מימוש פולימורפי – כתיבת התכונות של כל **קטגוריה** בהתאם]

קטגוריית רכבי **Compact** תיכתב כך:
[צריכת דלק] - מספר עשרוני (double)
[יצרן] – מספר שלם (enum)

קטגוריית רכבי **Standard** תיכתב כך:
[תכולת מטען מקסימלית] - מספר עשרוני (double)
[יצרן] – מספר שלם (enum)

קטגוריית רכבי **Premium** תיכתב כך:
[האם חשמלי] – מספר שלם 0/1
[שדרוג] – מספר שלם (enum)
[יצרן] – מספר שלם (enum)

פורמט כתיבה של **מערך לקוחות** לקובץ:
[כמות לקוחות] – מספר שלם
[**לקוח** 1]
[**לקוח** 2]
...
[**לקוח** n]

כאשר כל **לקוח** ייכתב כך:
[שם פרטי] – מחרוזת
[שם משפחה] – מחרוזת
[תעודת זהות] – מחרוזת
[מספר טלפון] – מחרוזת
[גיל] – מספר שלם

פורמט כתיבה של **מערך השכרות** לקובץ:
[כמות השכרות] – מספר שלם
[**השכרה** 1]
[**השכרה** 2]

...
[**השכרה** n]

כאשר כל **השכרה** תיכתב כך:
[מספר סידורי] – מספר שלם
[מספר סידורי של כלי הרכב] – מספר שלם (בעת הקריאה מתבצע חיפוש כלי הרכב המתאים).
[תעודת זהות של לקוח] – מחרוזת (בעת הקריאה מתבצע חיפוש הלקוח המתאים).
[תאריך התחלת השכרה]
[תאריך סיום השכרה]
[עלות משוקללת להזמנה] – מספר עשרוני (float)
[פוליסת ביטוח]
[חשבונית]
[מספר סידורי של סניף האיסוף] – מספר שלם

כאשר **תאריך** ייכתב כך:
[יום] – מספר שלם
[חודש] – מספר שלם
[שנה] – מספר שלם

כאשר **פוליסת ביטוח** תיכתב כך לקובץ טקסט (ובאופן דחוס כפי שהוסבר מעלה לקובץ בינארי):
[מספר סידורי] – מספר שלם
[סוג הפוליסה] – מספר שלם (enum)
[עלות ליום] – מספר שלם

כאשר **חשבונית** תיכתב כך:
[מספר סידורי של חשבונית] – מספר שלם
[מספר סידורי של השכרה מתאימה] – מספר שלם
[עלות משוקללת להזמנה] – מספר עשרוני (float)
[תאריך ביצוע הזמנה]

פורמט כתיבה של **רשימת סניפים** לקובץ:
[כמות סניפים] – מספר שלם
[**סניף** 1]
[**סניף** 2]
...
[**סניף** n]

כאשר כל **סניף** ייכתב כך:
[כתובת]
[מספר סידורי של הסניף] – מספר שלם

כאשר כל **כתובת** תיכתב כך:
[עיר] – מחרוזת
[רחוב] – מחרוזת
[מספר רחוב] – מספר שלם
[מדינה] – מספר שלם (enum)