מגשימים
תכנית הסייבר הלאומית

עקרונות מתקדמים

תרגול כיתה שבוע 3

CARS

הקדמה

בתרגיל נבנה מחלקות שונות, שבסופו של דבר יהוו חניון מכוניות.

התרגיל נועד לעשות חזרה על נושאי שבועות 1,2 ולתרגל את הנושאים שהוצגו בשיעור האחרון.

את המחלקות של חלק 1,2 נבנה מההתחלה, כלומר לא יינתנו קבצי h.
מי שסיים/ה את חלק 2, יכול/ה להמשיך לחלק 3 שבו יסופקו הקבצים וחלק מהפונקציות כבר יהיו ממומשות בחלק 3 התפקיד שלכם/ן הוא להשלים את המתודות החסרות.

לא סיימנו? לא קרה כלום 😊,
קחו את התרגיל הביתה ותנסו לסיים לבד.

הוראות

קונבנציות וסדר:

את קוד התרגיל יש לכתוב על פי הקונבנציות של התכנית, מומלץ לעבור על מסמך הקונבנציות לפני שמתחילים לכתוב.

מחלקה יש לרשום באות גדולה.

פרקו את הקוד לפונקציות והקפידו על קוד מסודר לאורך התכנית.

שדות פרטיים יש לסמן בקו תחתון (לדוגמא name_)

הנחיות

לא ניתן להשתמש במבני נתונים של ספריית STL (בהמשך הקורס נלמד).

שימרו על עיקרון הכימוס (encapsulation), חשפו החוצה רק את הממשק המינימלי הדרוש (minimal API), הגדירו כמה שפחות שדות/מתודות כ- public.

הקפידו על קוד **מסודר ומתועד**.

השתמשו בהגדרת קבועים (#define) כדי להפוך את הקוד למסודר וקריא.

נסו לכתוב פונקציות יעילות ככל הניתן (זמן ריצה).

הימנעו מדליפות זיכרון.

זכרו לא לממש מתודות ללא צורך.

קובץ header - תזכורת:

קובץ header מכיל את הממשק של המחלקה.
בקובץ h אנו מגדירים את השדות והמתודות של המחלקה, בנוסף נקבע מי יכול לגשת לכל אחד מהם (public / private).
בקובץ h מופיעות החתימות של הפונקציות (מתודות) ללא המימוש שלהן.
בצורה דומה, מופיעות רק ההצהרות על המשתנים, ללא הצבת ערכים.

CarColor

חלק 1 – מחלקת CarColor:

בחלק זה נגדיר מחלקה חדשה שמייצגת צבע מכונית.

בתוך הקובץ CarColor.h נגדיר enum בשם Color אשר יכיל את כל הצבעים האפשריים.

*שימו לב שה - enum מוגדר באותו קובץ אבל לא מוגדר בתוך המחלקה עצמה.

לקריאה נוספת על enum

למחלקת CarColor ישנו שדה אחד מסוג Color (ה enum שהגדרנו)

- בנאי שמקבל את הצבע הרצוי (מסוג Color) – `CarColor(const Color& color);`
- בנאי ברירת מחדל - `CarColor();`
  מכונית תיהיה בצבע לבן אם לא נשלח צבע אחר.
- פונקציה שמחזירה את הצבע – `Color getColor() const;`.
- אופרטור השוואה '==' – אובייקטים של CarColor ייחשבו זהים אם הצבעים זהים.
  *הערה: על החתימות להיות מדויקות, חפשו באינטרנט איך נראות החתימות אם יש ספק.
- ב ++C אין דרך אלגנטית לתת ערך של מחרוזת ל enum, כלומר לא נוכל להגדיר לדוגמא enum Color{RED="red",BLUE="blue",} ולכן אם ברצוננו לאפשר הדפסה נוחה של enum נצטרך לעשות כמה מעקפים. תוכלו לקרוא על כמה מהם כאן. אין צורך לאפשר הדפסה של ה enum, אם תרצו תוכלו להוסיף זאת כמשימת בונוס 😊.

הוסיפו צבעים שברצונכם לאפשר.
ה - enum **חייב להכיל את הצבע RED (אדום)**

Car

חלק 2 – מחלקת Car:

שדות:

כל מכונית צריכה לכלול את התכונות הבאות:

שם משתנה	סוג	תיאור
_owner	std::string	שם בעל הרכב
_price	double	מחיר
_carColor	CarColor	צבע
_model	std::string	דגם
_company	std::string	יצרן

מתודות:

כל אובייקט של מכונית חייב לכלול את המתודות הבאות:

א. בנאי: מאתחל את כל השדות על פי הערכים שנשלחו כפרמטרים.
*הערה – את השדה CarColor עלינו לאתחל באמצעות שדה Color, ויש להשתמש ב initialization line על מנת לאתחל אותו.

ב. Getters/Setters: פונקציות המאפשרות גישה (קריאה או שינוי) של המשתנים הפרטיים של המחלקה. (זכרו להשתמש ב const וב reference)

ג. `void print() const;`
פונקציה אשר מדפיסה מכונית בפורמט הבא:
שימו לב שכל שדה מודפס בשורה נפרדת

```
"Model: <model>
 Company <company>
 Price: <price>$
 Owner <owner>"
```

אופרטורים:

הוסיפו תמיכה באופרטורים הבאים (היעזרו במידע על דריסת אופרטורים בקישור):

- **אופרטור <,>** - השוואה בין מכוניות באמצעות האופרטורים '>' ו- '<' תיעשה על פי המחיר של המכונית. לדוגמא בהינתן Car c1,c2 אם המחיר של c2 הוא 200 אלף, והמחיר של c1 הוא 100 אלף, הפקודה c1 < c2 תחזיר true, והפקודה c2 > c1 תחזיר true.
- **אופרטור** '==','=!' - השוואה ישירה בין מכוניות באמצעות האופרטור == תתבצע ע"פ שדות הדגם (model), החברה (company), הצבע (carColor), והמחיר (price). כלומר יכולות להיות שתי מכוניות עם בעלים שונים אשר נחשבות זהות ע"פ האופרטור שלנו.
  חשבו כיצד לממש את האופרטור =! (לא שווה) בשורה אחת בלבד.

(בונוס) - משתנים סטטיים:

קראו באינטרנט על משתנים סטטיים, לימדו כיצד מגדירים משתנה סטטי ב ++C. בהמשך הקורס נלמד על משתנים ופונקציות סטטיות בצורה יותר מסודרת, אבל תוכלו לתרגל כבר עכשיו! 😎

הגדירו משתנה סטטי מסוג int בשם totalCars, אשר סופר את מספר המכוניות שנוצרו במהלך התכנית. המשתנה מאותחל כ- 0.

כתבו מתודה אשר מחזירה את ערכו.

CarNode

חלק 3 – מחלקת CarNode:

בחלק זה והחלק הבא תקבלו מימוש קיים למחלקה CarNode והמחלקה CarList המייצגת חוליה ברשימת מכוניות (אין צורך לממש בעצמנו), ורשימת מכוניות, להלן התיאור של המחלקות.

שדות:

כל חולייה כוללת את התכונות הבאות:

שם משתנה	סוג	תיאור
_data	Car	המכונית שמוחזקת ע"י החוליה
_next	CarNode*	החולייה הבאה בתור

מתודות:

כל חולייה כוללת את המתודות הבאות:

א. בנאי: מאתחל חולייה חדשה עם המידע שנשלח
החולייה תחילה תצביע על nullptr, וה - data שלה יהיה המכונית ששלחו.

ב. Getters/Setters: פונקציות המאפשרות גישה (קריאה או שינוי) של המשתנים הפרטיים של המחלקה.

CarList

חלק 4 – מחלקת CarList:

להלן תיאור המחלקה שתהווה חניון מכוניות.

שדות:

רשימת מכוניות כוללת את התכונות הבאות:

שם משתנה	סוג	תיאור
_first	CarNode*	החולייה הראשונה ברשימה
_size	int	גודל הרשימה (מספר החוליות)

מתודות:

המתודות הבאות מומשו עבורכם/ן, אין צורך לכתוב אותן.

א. בנאי: מאתחל רשימת מכוניות חדשה. (אין פרמטרים)
החולייה הראשונה תצביע על nullptr, גודל הרשימה 0.

ב. בנאי העתקה ואופרטור השמה: מעתיק תוכן מכונית אחת אל מכונית אחרת. חישבו האם מספיק להשתמש בהעתקה רדודה (shallow copy), או שיש צורך לבצע העתקה עמוקה (deep copy).
רמז: זכרו שבכל חוליה יש מצביע.

ג. Getters/Setters: פונקציות המאפשרות גישה (קריאה או שינוי) של המשתנים הפרטיים של המחלקה.

ד. `bool find(const Car& car) const;`
הפונקציה תחזיר true במידה והמכונית שנשלחה נמצאת ברשימה (אחת החוליות מחזיקה אותה).

ה. `void add(const Car car)`
תוסיף חולייה חדשה לסוף הרשימה, על החולייה להחזיק את המכונית שנשלחה כפרמטר.

ו. `bool remove(const Car& car);`
תסיר את החולייה אשר מחזיקה במכונית שנשלחה כפרמטר. במידה ויש כמה חוליות אשר מחזיקות מכונית כזו הפונקציה תסיר את הראשונה שתפגוש. במידה ולא ניתן היה למצוא את המכונית ברשימה, הפונקציה תחזיר false, אם החולייה הוסרה בהצלחה הפונקציה תחזיר true.

ממשו את המתודות הבאות בעצמכם/ן:

```
void mostExpensive() const;
```

תדפיס את המכונית היקרה ביותר ברשימה. (יש להשתמש באופרטורים שמימשתם/ן)

```
int numOfRedCars() const;
```

תחזיר את מספר המכוניות האדומות ברשימה

```
int numOfOwnerCars(const std::string owner) const;
```

תחזיר את מספר המכוניות אשר בבעלות בעל הרכב שנשלח

```
void print() const;
```

תדפיס את כל המכוניות ברשימה

```
void printByModule(const char c) const;
```

תדפיס את כל המכוניות ברשימה ששם המודל שלהם מתחיל באות שנשלחה

בנו main אשר בודק את תקינות המתודות.

בהצלחה!!!!