234124 - מבוא לתכנות מערכות
תרגיל בית מספר 3 (C++)

סמסטר חורף תש״פ

אחראים : רומן שפירא romanshap@cs.technion.ac.il

תאריך הגשה : 26.1.20

אופן הגשה : בזוגות, הגשה אלקטרונית בלבד, באמצעות האתר של הקורס.

פרטים נוספים הרלוונטיים להגשה רשומים בסוף תרגיל זה.

משקל התרגיל : 12%

1 הערות כלליות

- את התרגיל יש לכתוב ולהגיש **בזוגות**.
- לשאלות בנוגע להבנת התרגיל יש לשאול בפורום של הקורס במודל, או בשעות הקבלה של אחד המתרגלים.
- כל חומר נלווה לתרגיל נמצא על השרת בתיקייה ~mtmchk/public/1920a/ex3.
- קראו את התרגיל עד סופו לפני שאתם מתחילים לממש. ייתכן שתצטרכו להתאים את המימוש שלכם לחלק עתידי בתרגיל. תכננו את המימוש שלכם לפני שאתם ניגשים לעבוד.
- מומלץ מאוד לכתוב את הקוד בחלקים קטנים, לקמפל כל חלק בנפרד על השרת, ולבדוק שהוא עובד באמצעות שימוש בטסטים (אין צורך להגיש טסטים כנ"ל). אנו מעודדים אתכם לחלוק טסטים עם חבריכם.
- העתקות קוד בין סטודנטים תטופלנה בחומרה! אף על פי כן, מומלץ ומבורך להתייעץ עם חברים על ארכיטקטורת המימוש.
- שימו לב: **לא יינתנו דחיות במועד התרגיל, פרט למקרים מיוחדים**. תכננו את הזמן בהתאם.
- הארכה במועד ההגשה תינתן לסטודנטים שמבצעים שירות מילואים (בעבור כל יום מילואים יקבלו זוג הסטודנטים חצי יום נוסף להגשה, נתחשב כמובן מעבר כתלות במקרה המדובר).

2 חלק א – מימוש מבנה נתונים גנרי

2.1 תיאור כללי

בחלק זה של התרגיל נרצה לממש מבנה נתונים גנרי ב-C++ תוך שימוש בתבניות (Templates) וחריגות.

<u>הערה:</u> כיוון שבמועד פרסום התרגיל טרם נלמדו תבניות וחריגות (יילמדו בתרגול 10), מומלץ בתור התחלה לממש את הUniqueArray בצורה לא גנרית וללא חריגות (למשל עבור int בלבד) ולהפוך את המימוש שלכם לגנרי/להוסיף חריגות לאחר לימוד הנושאים.

מבנה הנתונים אותו נממש נקרא **UniqueArray**, והוא שילוב של מערך רגיל ו-Set שראיתם במהלך הסמסטר:

1. ל-UniqueArray כמות מקסימלית של איברים הנקבעת עם יצירתו (ולא משתנה עד הריסתו), בדומה למערך.

2. הוספת איברים מתבצעת ע"י פעולת insert בדומה ל-Set. כל איבר המוכנס ל-UniqueArray מקבל מספר סידורי (אינדקס, בדומה למערך), אשר מוחזר כאשר האיבר מוכנס בהצלחה. מספר זה אינו משתנה כל עוד האיבר נמצא ב-UniqueArray. אם מנסים להכניס איבר שכבר קיים יוחזר האינדקס של האיבר הקיים (כל איבר יכול להופיע פעם אחת בלבד בדומה ל-Set).
3. ניתן לקבל את המספר הסידורי של איבר ב UniqueArray-ע"י פעולת.getIndex
4. נרצה לממש פעולת filter על ה-UniqueArray, שתחזיר UniqueArray חדש בו יופיעו רק האיברים העומדים בפילטר.

כאמור ה-UniqueArray ישתמש בתבניות. נרצה לספק לUniqueArray שני פרמטרי תבנית:

1. Class Element – זהו טיפוס האלמנטים של ה-UniqueArray. **שימו לב**: אסור להניח כי ל-Element יש בנאי חסר פרמטרים.
2. Class Compare – זוהי פונקציה (או Function Object) אשר מממשת השוואה בין שני אלמנטים ב-UniqueArray (נשתמש כפרמטר ברירת מחדל ב- std::equal_to<Element> אשר מממש את ההשוואה הרגילה בין שני משתנים מטיפוס Element. ניתן למצוא תיעוד מדויק של std::equal_to בקישור הבא: https://en.cppreference.com/w/cpp/utility/functional/equal_to).

2.2 ממשק ה- *UniqueArray*

הפעולות אותן עליכם לממש עבור ה-UniqueArray הינן:

1. בנאי ליצירת UniqueArray:

```
UniqueArray(unsigned int size);
```

פרמטרים: size – הגודל המקסימלי של ה-UniqueArray

2. בנאי העתקה ל-UniqueArray:

```
UniqueArray(const UniqueArray& other);
```

פרמטרים: other – ה-UniqueArray אותו יש להעתיק

3. הורס ה-UniqueArray:

```
~UniqueArray();
```

4. הוספת איבר ל-UniqueArray:

```
unsigned int insert(const Element& element);
```

פרמטרים: element – האלמנט אותו יש להוסיף ל-UniqueArray

ערך חזרה: - מספר סידורי פנוי כלשהו (במידה והאלמנט קיים יוחזר המספר הסידורי הקיים)

חריגות: UniqueArrayIsFullException – במקרה והUniqueArray מלא ולא ניתן להוסיף את האיבר

5. פעולת החזרת המספר הסידורי של אלמנט ב-UniqueArray:

```
bool getIndex(const Element& element, unsigned int& index)
const;
```

פרמטרים: element – האלמנט ב-UniqueArray עבורו יש להחזיר את המספר הסידורי

index – רפרנס למשתנה אליו יש להחזיר את המספר הסידורי.

ערך חזרה: - true אם האלמנט נמצא, false אחרת. במידה והאלמנט נמצא מספרו הסידורי יושם לתוך index.

6. אופרטור [] להחזרת אלמנט מתוך ה-UniqueArray:

```
const Element* operator[] (const Element& element) const;
```

פרמטרים: element – האלמנט ב-UniqueArray אותו יש להחזיר.

ערך חזרה: - מצביע לאלמנט ב-UniqueArray השווה לElement הנתון (לפי ה-Compare הנתון בתבנית). אם האלמנט לא קיים יוחזר מצביע NULL.

7. הסרת אלמנט מה-UniqueArray:

```
bool remove(const Element& element);
```

פרמטרים: element – האלמנט אותו יש להסיר מה-UniqueArray

ערך חזרה: - true אם האלמנט הוסר, false אחרת

8. פעולה להחזרת מספר האלמנטים הנוכחי ב-UniqueArray:

```
unsigned int count() const;
```

פרמטרים: -

ערך חזרה: מספר האלמנטים הנוכחי ב-UniqueArray

9. פעולה להחזרת מספר האלמנטים המקסימלי ב-UniqueArray:

```
unsigned int size() const;
```

פרמטרים: -

ערך חזרה: מספר האלמנטים המקסימלי ב-UniqueArray.

10. פעולת filter:

```
UniqueArray<Element, Compare> filter(const Filter& f) const;
```

פרמטרים: f – משתנה מטיפוס Filter המהווה טיפוס בסיס לFunctor שמקבל const Element& ומחזיר bool (ראו סעיף 2.3 לפירוט)

ערך חזרה: מחזירה UniqueArray<Element, Compare> חדש בגודל זהה לנוכחי, אשר מכיל רק את האיברים מה-UniqueArray הנוכחי (this) שעבורם f החזירה true. שימו לב שהאינדקסים של האיברים ב-UniqueArray החדש יהיו זהים לאינדקסים ב-UniqueArray הנוכחי.

2.3 דגשים למימוש

1. תוגדר מחלקת עזר פנימית UniqueArrayIsFullException עבור הניסיון להוסיף איבר לUniqueArray מלא.
2. בקובץ UniqueArray.h הנתון לכם, מופיעה הכרזת מחיקה של ה- copy-assignment operator:

```
UniqueArray& operator=(const UniqueArray&) = delete;
```

אם במימוש שלכם יש צורך באופרטור, אתם רשאים למחוק את הdelete= ולממשו. אחרת, אין למחוק שורה זו.

3. תוגדר מחלקת עזר פנימית Filter המהווה טיפוס בסיס אבסטרקטי לאובייקט פונקציה שיממש את אופרטור () הבא:

```
virtual bool operator() (const Element&) = 0;
```

כלומר זוהי פונקציה הבודקת קיום תנאי כלשהו על אלמנט של ה-UniqueArray.

4. את ההכרזות עליכם לכתוב בקובץ UniqueArray.h (אשר מסופק לכם חלקית). ניתן להוסיף פעולות נוספות כראות עיניכם עם תיעוד מתאים.
5. את המימוש עליכם לכתוב בקובץ UniqueArrayImp.h לו מבצעים include בתוך UniqueArray.h. **שימו לב: אין לבצע "include "UniqueArray.h בקובץ UniqueArrayImp.h!**
6. אין להשתמש בSTL עבור מימוש ה-UniqueArray (פרט לברירת מחדל עבור פרמטר התבנית Compare)
7. הינכם יכולים להוסיף פעולות ממשק נוספות כראות עיניכם עבור המשך התרגיל – אך כמובן שיש לשמור על ה-Encapsulation של המבנה.
8. עליכם להניח שהמימוש שלכם יעבור בדיקות יחידה עבור כל אחת מהפעולות המוגדרות. מסופק לכם קובץ בדיקה UniqueArrayTests.cpp שבודק התנהגות תקינה בסיסית של הUniqueArray, אך אין זה תחליף לבדיקות יחידה מקיפות ולכן אנו מעודדים אתכם לבנות אותן בעצמכם כדי לוודא תקינות.

3 חלק ב – שאלות יבשות על ה-UniqueArray

עליכם לענות על השאלות הבאות ולהגישן בקובץ PDF בשם dry.pdf שיצורף ל-zip ההגשה הסופית.

1. אילו דרישות צריכות להתקיים ע"י פרמטרי התבנית Element ו-Compare בהם אנו משתמשים ב-UniqueArray?
2. נניח כי היינו מגדירים ע"י operator overloading את האופרטור הבא, המקבל מספר סידורי של איבר ב-UniqueArray ומחזיר את המצביע לאיבר:

```
const Element* operator[] (unsigned int index) const;
```

איזו בעיה עלולה להיווצר כעת בשימוש ב-UniqueArray? כיצד ניתן להתגבר על הבעיה?

את הפתרון יש להגיש בקובץ PDF בשם dry.pdf, שיצורף ל-zip של החלק הרטוב. מומלץ להשתמש בתוכנת Word כדי לכתוב את הפתרון, ואז להשתמש ב-File->Export על מנת לייצא לקובץ PDF.

4 חלק ג – מימוש מערכת למגרש חניה MtmParkingLot

4.1 תיאור כללי

בחלק זה של התרגיל נממש מערכת עבור מגרש חניה. מטרת המערכת היא מעקב אחר כניסה/יציאה של רכבים למגרש, הדפסת מידע בעת הצורך, וטיפול בתשלום בהתאם לזמן החניה וסוג הרכב.

- ישנם 3 סוגי רכבים הנתמכים במערכת, והם יכולים לחנות באיזורי חניה ספציפיים ("בלוקים") בהתאם לסוג הרכב:
  1. אופנועים יכולים לחנות בבלוק חניות לאופנועים בלבד
  2. רכב פרטי רגיל יכול לחנות בבלוק לרכבים פרטיים בלבד
  3. רכב של אדם נכה יכול לחנות בבלוק חניית נכים (בעדיפות ראשונה), ובמידה והיא מלאה גם בבלוק של רכבים פרטיים.

- מספר החניות בכל בלוק חניה (עבור אופנוע/רכב רגיל/חניות נכים) נקבע מראש בעת יצירת מבנה מגרש החניה
- תשלום של כל רכב נקבע לפי סוג הרכב וזמן החניה, בהתאם לחוקים הבאים:
  - אופנוע ישלם 10 ש"ח עבור שעת חניה ראשונה או חלק ממנה, ו-5 ש"ח עבור כל שעת חניה נוספת או חלק ממנה, וישלם לכל היותר על 6 שעות. לדוגמא:
    - אופנוע שחנה במשך 3 דקות ישלם 10 ש"ח
    - אופנוע שחנה במשך שעה (60 דקות) ישלם 10 ש"ח
    - אופנוע שחנה במשך שעה ודקה ישלם 15 ש"ח
    - אופנוע שחנה במשך 15 שעות ישלם 35 ש"ח (מחיר של 6 שעות)
  - רכב פרטי רגיל ישלם כמו אופנוע, אך 20 ש"ח עבור שעה ראשונה ו-10 ש"ח עבור שעות נוספות (ולכל היותר על 6 שעות).
  - רכב נכה ישלם 15 ש"ח קבוע, ללא תלות בזמן החניה. לדוגמא:
    - רכב נכה שחנה במשך 3 דקות ישלם 15 ש"ח
    - רכב נכה שחנה במשך 5 שעות ישלם 15 ש"ח
    - שימו לב שהמחיר עבור רכב נכה לא תלוי באם הוא חנה בחניית נכים או בחניית רכב רגיל.
- חניית כל הרכבים במגרש החניה מוגבלת ל-24 שעות מרגע כניסתם. מדי פעם מגיע למגרש החניה פקח, כך שכל הרכבים שחונים יותר מ-24 שעות יקבלו קנס של 250 ש"ח, אותו ישלמו בזמן היציאה בנוסף לתשלום הרגיל על החניה. שימו לב שהקנס אינו תלוי בסוג הרכב. כמו כן כל רכב יכול להיקנס לכל היותר פעם אחת.
- על המערכת לתמוך בפעולות של:
  1. כניסת רכב למגרש החניה
  2. יציאה ותשלום של רכב ממגרש החניה
  3. הגעת הפקח למגרש החניה
  4. הדפסת כל הרכבים במגרש החניה

4.2 מבנה המערכת

בחלק זה של התרגיל עליכם לממש את מחלקת ParkingLot שתיאורה מובא מיד. במידת הצורך רצוי (ומומלץ) להגדיר מחלקות עזר נוספות. כמו כן עומדים לרשותכם מספר טיפוסים/מחלקות/קבועים אותן הגדיר סגל הקורס, שתיאורן יובא בהמשך.

4.2.1 מחלקת ParkingLot

זוהי מחלקה המייצגת את המבנה של מגרש החניה. עליכם לממש עבור מגרש החניה את הפעולות הבאות:

1. בנאי ליצירת ParkingLot חדש:

```
ParkingLot(unsigned int parkingBlockSizes[]);
```

<u>פרמטרים:</u> מערך של מספר מקומות החניה לכל סוג רכב, כאשר סדר האיברים לפי הערכים המספריים של VehicleType (כלומר Motorbike=0, HandicappedCar=1, Car=2)

2. הורס ParkingLot:

```
~ParkingLot();
```

<u>פרמטרים:</u> -

3. פעולה להכנסת רכב למגרש החניה, בהתאם לחוקים ב-4.1, והדפסת הודעה מתאימה:

```
ParkingResult enterParking(VehicleType vehicleType,
LicensePlate licensePlate, Time entranceTime);
```

פרמטרים: vehicleType – סוג הרכב הנכנס

licensePlate – לוחית רישוי של הרכב הנכנס

entranceTime – זמן הכניסה של הרכב

ערך חזרה ופעולה: יוחזר משתנה מטיפוס ParkingResult המוגדר כחלק מהטיפוסים הנתונים (ראו 4.3 לפרטים), ויודפסו הודעות בהתאם לחוקיות הבאה:

a. עבור כניסה מוצלחת, תתבצע הדפסה של הרכב ע"י מתודת ParkingLotPrinter::printVehicle והדפסת כניסה מוצלחת ע"י מתודת ParkingLotPrinter::printEntrySuccess (בסדר הזה). המתודות מוגדרות במחלקת ParkingLotPrinter המסופקת לכם (ראו 4.3). יוחזר SUCCESS.
b. כאשר אין מקום חניה לרכב, תתבצע הדפסה של הרכב והדפסת כניסה לא מוצלחת ע"י מתודת ParkingLotPrinter::parkingEntryFailureNoSpot. יוחזר NO_EMPTY_SPOT.
c. כאשר הרכב המוכנס כבר נמצא במגרש החניה תתבצע הדפסה של הרכב (הקיים) והדפסה ע"י מתודת ParkingLotPrinter::parkingEntryFailureAlreadyParked. יוחזר ALREADY_PARKED.

4. פעולה להוצאת רכב ממגרש החניה, והדפסת הודעה מתאימה:

```
ParkingResult exitParking(LicensePlate licensePlate, Time
exitTime);
```

פרמטרים: licensePlate – לוחית רישוי של הרכב היוצא

exitTime – זמן היציאה של הרכב

ערך חזרה ופעולה: יוחזר משתנה מטיפוס ParkingResult המוגדר כחלק מהטיפוסים הנתונים (ראו 4.3 לפרטים), ויודפסו הודעות בהתאם לחוקיות הבאה:

a. עבור יציאה מוצלחת של הרכב, תתבצע הדפסה של הרכב ע"י ParkingLotPrinter:printVehicleוהדפסת יציאה מוצלחת ע"י מתודת ParkingLotPrinter::parkingExitSuccess.יוחזרSUCCESS.
b. עבור יציאה לא מוצלחת של הרכב (הרכב לא נמצא במגרש החניה), תודפס הודעה ע"י מתודת ParkingLotPrinter::parkingExitFailure. יוחזר VEHICLE_NOT_FOUND.

5. פעולה לקבלת מקום החניה של רכב:

```
ParkingResult getParkingSpot(LicensePlate licensePlate,
ParkingSpot& parkingSpot) const;
```

פרמטרים: licensePlate – לוחית הרישוי של הרכב

parkingSpot – משתנה ParkingSpot אליו יש להחזיר את חניית הרכב במידה והרכב נמצא במגרש החניה.

ערך חזרה: ParkingResult בהתאם לתוצאה:

a. SUCCESS – במידה והרכב נמצא, מיקום החניה שלו יושם לתוך משתנה parkingSpot.
b. VEHICLE_NOT_FOUND – במידה והרכב בעל לוחית הרישוי לא נמצא במגרש.

6. אופרטור הדפסה של מגרש החניה:

```
friend ostream& operator<<(ostream& os, const ParkingLot&
parkingLot);
```

פרמטרים: os – הstream לתוכו יש להדפיס

parkingLot – מגרש החניה אותו נרצה להדפיס

ערך חזרה: os – הstream לאחר ההדפסה.

אופן פעולה: תחילה יש לקרוא למתודת ParkingLotPrinter::printParkingLotTitle על מנת להדפיס כותרת, לאחר מכן עבור כל רכב יש לבצע:

a. הדפסת רכב ע"י מתודת ParkingLotPrinter::printVehicle
b. קריאה למתודת ParkingLotPrinter::printParkingSpot, המקבלת כפרמטר משתנה מטיפוס ParkingSpot המוגדר לכם (ראו 4.3 לפרטים). שימו לב שמתודה זו כבר כוללת ירידת שורה כך שאין צורך לכלול ירידת שורה נוספת.

סדר הדפסת הרכבים יהיה ממוין לפי ה-ParkingSpot שלהם. ניתן להשתמש ב-STL עבור המיון (שימו לב שלמחלקת ParkingSpot כבר מוגדר אופרטור השוואה).

7. פעולת ביקור הפקח:

```
void inspectParkingLot(Time inspectionTime);
```

פרמטרים: inspectionTime – שעת הגעת הפקח

אופן פעולה: יש למצוא את כל הרכבים בחניון החונים יותר מ-24 שעות בזמן ביקור הפקח. עבור כל רכב כזה יש לסמנו כך שישלם קנס של 250 ש"ח הנוסף על התשלום הרגיל בזמן היציאה. כמו כן תתבצע קריאה למתודת ParkingLotPrinter::printInspectionResult המקבלת כפרמטר את זמן ביקור הפקח ומספר הרכבים שנקנסו ומדפיסה הודעה מתאימה.

4.2.2 דגשים לגבי המימוש

- נתון לכם קובץ Header עבור מחלקת ParkingLot, עליכם להשלים אותו ולהוסיף קובץ cpp מתאים.
- אלא אם נאמר אחרת, מותר לכם להוסיף פעולות/מחלקות עזר, אך כמובן שיש לשמור על Encapsulation ולתעד כל מחלקה/פעולה אותה אתם מוסיפים.
- עבור הדפסות, אין לשנות את פורמט ההדפסה. בפרט אין צורך להוסיף הדפסות של מחרוזות משלכם (כולל רווחים/תווי סוף שורה). יש להשתמש בפונקציות שנתונות לכם במחלקת ParkingLotPrinter.
- חישבו היטב על איך לייצג את הרכבים ב-ParkingLot, כיצד להקצות להם ParkingSpot וכיצד לממש את חוקי כניסת הרכבים.
- מותר ורצוי להשתמש ב-UniqueArray שהגדרתם בחלק א.
- מותר להשתמש בSTL בחלק זה של התרגיל.
- ניתן להניח שהקריאה למתודות ה-ParkingLot מתבצעת בסדר כרונולוגי – כלומר לא ייתכן שנקרא לexitParking עבור רכב עם זמן יותר קטן מזמן כניסתו.
- יש לממש את כל המחלקות שלכם תחת namespace MtmParkingLot
- עליכם להניח שהמימוש שלכם יעבור בדיקות יחידה עבור כל אחת מהפעולות המוגדרות (מלבד בדיקות קלט/פלט המסופקות לכם עבור תכנית MtmParkingLot).

4.3 הגדרות וטיפוסים נתונים

להלן פירוט ההגדרות של הטיפוסים הנתונים לכם איתם אתם יכולים לממש את המערכת. אין צורך לממש מחדש טיפוסים אלו!

כל הטיפוסים המוגדרים בסעיפים הבאים נמצאים תחת namespace ParkingLotUtils.

שימו לב! אין לשנות את הטיפוסים/מחלקות/קבועים הנתונים לכם. אתם לא מגישים קבצים אלו והקוד שלכם יקומפל עם עותק של הקבצים האלו מהשרת.

4.3.1 מחלקת Time

זוהי מחלקה הממשת ADT של זמן עבור מערכת מגרש החניה, בה הינכם צריכים להשתמש כדי לקבל זמני כניסה ויציאה של רכבים ולחשב זמני חניה ועלויות. תיאור הפעולות נמצא בקובץ Time.h המסופק לכם.

4.3.2 מחלקת ParkingSpot

זוהי מחלקה הממשת ADT של מיקום חניה (בלוק + מס' סידורי). עליכם להשתמש במחלקה זו על מנת להחזיר מיקום חניה של רכב במתודת getParkingSpot, וכמו כן עליכם להדפיס את מגרש החניה ממוין לפי ה-ParkingSpot של כל רכב. תיאור הפעולות נמצא בקובץ ParkingLot.h המסופק לכם.

4.3.3 מחלקת ParkingLotPrinter

זוהי מחלקת עזר המספקת לכם פעולות סטטיות עבור כל ההדפסות במערכת. תיאור הפעולות נמצא בקובץ ParkingLotPrinter.h המסופק לכם.

4.3.4 קובץ ParkingLotTypes.h

זהו קובץ header המכיל הגדרות טיפוסים נוספים אותם אתם תצטרכו עבור התרגיל (שאינם מחלקות). ניתן לעשות Include לקובץ זה בכל מקום בקוד בו יש צורך. הקובץ מכיל את ההגדרות הבאות:

1. הגדרת טיפוס LicensePlate עבור לוחית הרישוי של כל רכב (מוגדר להיות string).
2. Enum ParkingResult – מגדיר תוצאות אפשריות של פעולות על מגרש החניה:
   a. SUCCESS – הצלחת הפעולה
   b. NO_EMPTY_SPOT – אין מקום חניה לרכב
   c. VEHICLE_NOT_FOUND – הרכב לא נמצא במגרש החניה
   d. VEHICLE_ALREADY_PARKED – הרכב כבר חונה במגרש החניה
3. Enum VehicleType – מגדיר את סוגי הרכבים האפשריים במערכת:
   a. MOTORBIKE – אופנוע
   b. HANDICAPPED – רכב נכה
   c. CAR – רכב רגיל
   d. FIRST – מכיל את הערך של סוג הרכב הראשון בטיפוס (MOTORBIKE)
   e. LAST – מכיל את הערך של סוג הרכב האחרון בטיפוס (CAR)

4.3.5 קובץ MtmParkingLot.cpp

קובץ זה מכיל את המעטפת לתכנית. בקובץ זה קיימת פונקציית ה-main של התכנית ותפקידו לקבל קלט מהמשתמש (בין אם בקובץ/הקלדה לתוך cin) ולהשתמש במחלקת ParkingLot שמימשתם על מנת לסמלץ את מגרש החניה. השימוש בתכנית יהיה מהצורה הבאה:

```
./MtmParkingLot [inputFile]
```

אופן פעולת התכנית הוא כלהלן:

1. פתיחת ה-input stream – כלומר במידה והמשתמש נתן קובץ קובץ קלט אז נפתח אותו לקריאה, אחרת נשתמש ב-cin.
2. אתחול הזמן הנוכחי (ל-0) ואתחול מגרש החניה

3. מעבר על פקודות המשתמש
4. סגירת ה-input stream (במידת הצורך) וסיום התכנית. כדי לצאת מהתכנית כאשר אנו קולטים קלט מ-cin יש לכתוב EOF (Ctrl+D), אחרת התכנית תסיים לרוץ לאחר שכל הפקודות מהקובץ בוצעו (הגענו לEOF בקובץ הקלט).

הפקודות המותרות בתכנית הינן:

1. ENTER <VehicleType> <LicensePlate> - פקודה להכנסת רכב בעל לוחית רישוי < License Plate> למגרש חניה. <Vehicle Type> יכול לקבל אחד מהערכים: { Motorbike, Handicapped, Car}.
2. EXIT <LicensePlate> - פקודה להוצאת רכב ממגרש החניה
3. INSPECT – פקודה לביקור הפקח
4. PRINT – פקודה להדפסת מגרש החניה
5. PASS_TIME <# of mins> - מזיזה את הזמן הנוכחי קדימה במספר הדקות הנתון.

לאחר שסיימתם לממש את המחלקות שעליכם לממש, יש לקמפל את התכנית ולהריץ אותה על קבצי הקלט הנתונים לכם תחת io_files/input*.txt. הפלט המצופה אמור להתאים לקובץ io_files/output*.txt (למעט מקרים מסוימים, ראו הערה למטה). קיימת אפשרות לבדוק את ההתאמה בצורה אוטומטית ע"י סקריפט שמסופק לכם (final_check.py פרטים בסעיף 5). אנו מעודדים אתכם לא להסתפק בקבצים אלו בלבד ולהוסיף בדיקות נוספות בעצמכם.

<u>הערה:</u> עקב חופש הבחירה שיש בידיכם במימוש הקצאת המספר הסידורי ב UniqueArray והקצאת מספרי החניות לרכבים ב-ParkingLot יתכנו הבדלים בין הפלט של התכנית שלכם לקבצי הפלט המסופקים - במספרי החניה של הרכבים ובסדר הדפסתם בפעולת Print. הבדלים כאלה לא בהכרח מעידים כי המימוש שלכם אינו נכון, והמימוש בכל מקרה ייבדק בבדיקות המתייחסות למקרים האלו (מטרת קבצי הקלט/פלט היא רק לתת לכם הערכה לנכונות המימוש). אנו מעודדים אתכם לבדוק את המימוש שלכם בבדיקות יחידה משלכם ולא להסתמך על בדיקות הקלט/פלט בלבד.

4.4 דרישות נוספות לחלק הרטוב

4.4.1 Makefile

עליכם לספק Makefile כמו שנלמד בקורס עבור בניית הקוד של חלק ב של התרגיל. הכללים המינימליים שצריכים להופיע ב-Makefile הינם:

- כלל MtmParkingLot שיבנה את התכנית MtmParkingLot
- כלל לכל ADT שהוספתם בחלק ב של התרגיל
- כלל clean אשר מוחק את כל תוצרי הקימפול

4.4.2 הידור

התרגיל ייבדק על שרת csl3 ועליו לעבור הידור בעזרת הפקודה הבאה:

```
g++ -std=c++11 -Wall -Werror -pedantic-errors –DNDEBUG -o
MtmParkingLot *.cpp
```

כמו כן ייבדק בנפרד מימוש ה-UniqueArray, עליו לעבור הידור בעזרת הפקודה הבאה:

```
g++ -std=c++11 -Wall -Werror -pedantic-errors -g
tests/UniqueArrayTest.cpp -o UniqueArrayTest
```

עליכם לוודא שהרצה של פקודות אלו על csl3 אכן יוצרת את התוכניות הנדרשות מכם.

4.4.3 ולגרינד ודליפות זיכרון

המערכת חייבת לשחרר את כל הזיכרון שעמד לרשותה בעת ריצתה. על כן עליכם להשתמש ב-valgrind שמתחקה אחר ריצת התוכנית שלכם, ובודק האם ישנם משאבים שלא שוחררו. הדרך לבדוק האם יש לכם דליפות בתוכנית היא באמצעות שתי הפעולות הבאות (שימו לב שחייב להיות main, כי מדובר בהרצה ספציפית):

1. קימפול של השורה לעיל עם הדגל g-
2. הרצת השורה הבאה:

```
valgrind --leak-check=full ./MtmParkingLot
```

כאשר MtmParkingLot זה שם קובץ ההרצה.

הפלט ש-valgrind מפיק אמור לתת לכם, במידה שיש לכם דליפות, את שרשרת הקריאות שהתבצעו שגרמו לדליפה. אתם אמורים באמצעות דיבוג להבין היכן היה צריך לשחרר את אותו משאב שהוקצה ולתקן את התוכנית. בנוסף, valgrind מראה דברים נוספים כמו קריאה לא חוקית (למשל קריאה לזיכרון שכבר שוחרר) – גם שגיאות אלו עליכם להבין מהיכן מגיעות ולתקן.

4.4.4 בדיקת התרגיל

התרגיל ייבדק בדיקה יבשה (מעבר על קונבנציות הקוד והארכיטקטורה) ובדיקה רטובה.

הבדיקה היבשה כוללת מעבר על הקוד ובודקת את איכות הקוד (שכפולי קוד, קוד מבולגן, קוד לא ברור, שימוש בטכניקות תכנות "רעות").

הבדיקה הרטובה כוללת את הידור התוכנית המוגשת והרצתה במגוון בדיקות אוטומטיות. על מנת להצליח בבדיקה שכזו, על התוכנית לעבור הידור, לסיים את ריצתה, ולתת את התוצאות הצפויות.

5 אופן ההגשה

את ההגשה יש לבצע דרך אתר הקורס, תחת Assignments -> HW3 -> Electronic Submit. הקפידו על הדברים הבאים:

- יש להגיש את קבצי הקוד מכווצים לקובץ zip (לא פורמט אחר), כאשר כל הקבצים מופיעים בתיקיית השורש בתוך קובץ ה-zip.
- **יש להגיש אך ורק את קבצי ה-h וה-cpp אשר נדרשתם לכתוב/לשנות. אין להגיש את הקבצים אשר סופקו לכם.**
- הקבצים אשר מסופקים לכם יצורפו על ידינו במהלך הבדיקה. בפרט, ניתן להניח את קיום קבצי ParkingLotPrinter.cpp, ParkingLotPrinter.h, Time.cpp, Time.h, ParkingSpot.cpp ParkingSpot.h, ParkingLotTypes.h בתקייה הראשית, את קיום תקיית tests עם קובץ UniqueArrayTest.cpp בתוכה, ואת קיום תקיית io_files עם (לפחות) הקבצים הנתונים לכם בתוכה.
- ניתן להגיש את התרגיל מספר פעמים, רק ההגשה האחרונה נחשבת.
- על מנת לבטח את עצמכם נגד תקלות בהגשה האוטומטית, שמרו את קוד האישור עבור ההגשה. עדיף לשלוח גם לשותף. כמו כן שמרו עותק של התרגיל של חשבון ה-csl3 שלכם לפני ההגשה האלקטרונית ואל תשנו אותו לאחריה (שני הקובץ יגרור שינוי חתימת העדכון האחרון).
  - כל אמצעי אחר לא ייחשב הוכחה לקיום הקוד לפני ההגשה.

לנוחותכם, אנו מספקים סקריפט בשם finalcheck.py לשימושכם לצורך וידוא תקינות ההגשה. הסקריפט מוודא שה-zip מכיל רק את הקבצים הנדרשים, ומנסה להדר את הקוד ולהריץ אותו על קבצי הבדיקה הנתונים (כלומר מנסה להריץ את UniqueArrayTest וכמו כן על הקבצים תחת io_files). להרצת הסקריפט, הריצו את השורה הבאה (כאשר ex3_sol.zip הוא ה-zip שאתם עומדים להגיש):

```
~mtmchk/public/1920a/ex3/finalcheck.py ex3_sol.zip
```

זכרו, הסקריפט הוא לצורכי נוחות בלבד, וזו עדיין אחריותכם לוודא שההגשה עומדת בכל התנאים.

בהצלחה!