מספר זהות:

סמסטר א' תשע"ב
מועד: ב' 9/3/2012
משך הבחינה: 3½ שעות
חומר עזר: כל חומר עזר כתוב מותר

בחינה בקורס: תכנות מכוון עצמים ושפת C++

מרצים: **אמיר קירש, ד"ר אלון שקלר**

מדבקת
ברקוד

הנחיות כלליות לבחינה:

- שימו לב, השאלון מודפס משני צדדיו.
- המבחן מורכב משני חלקים:
  חלק א' כולל 7 שאלות אמריקאיות. משקל כל שאלה 7 נקודות, סה"כ: 49 נק'.
  חלק ב' כולל שאלת תכנות שמשקלה הכולל 51 נק'.
- חובה לתעד בשאלת התכנות כל פעולה לא ברורה שנעשית.
- בשאלות האמריקאיות יש לסמן תשובה אחת לכל שאלה בטבלה המצורפת. במידה ומספר תשובות נראות נכונות יש לסמן את התשובה הנכונה ביותר.
- בסיום המבחן יש לרשום מס' ת.ז. על גבי טופס המבחן, לוודא שטבלת התשובות האמריקאיות נמצאת יחד עם טופס המבחן, ולהגישם בתוך מחברת הבחינה.
- המבחן הינו עם חומר פתוח. כל חומר עזר מותר למעט מכשירים אלקטרוניים למיניהם. אין להעביר חומר עזר בין תלמידים במהלך המבחן.
- נא לכתוב בכתב קריא ולא מחובר.

בהצלחה!

טבלת תשובות לחלק האמריקאי

חובה לספק הסבר עבור תשובה ו׳

	א	ב	ג	ד	ה	ו
שאלה 1						
שאלה 2						
שאלה 3						
שאלה 4						
שאלה 5						
שאלה 6						
שאלה 7						

הסברים לתשובות

חובה לספק הסבר במידה ונבחרה תשובה ו׳ (אף תשובה אינה נכונה).

מותר לצרף הסבר גם עבור תשובות אחרות. אמנם **רק** תשובה נכונה תזכה בניקוד עבור כל שאלה, אולם ניתן יהיה להסתמך על ההסבר במסגרת ערעור, אם יידרש. מומלץ לצרף הסבר לתשובה אמריקאית במיוחד במקרים בהם נראה לך שתשובתך דורשת הסבר או נימוק.

שאלה 1 :

שאלה 2 :

שאלה 3 :

שאלה 4 :

שאלה 5 :

שאלה 6 :

שאלה 7 :

חלק א' – שאלות אמריקאיות (49 נק' – 7 נק' לכל שאלה)

שאלות 1-3 מתייחסות לקטע הקוד הבא:

```
1.  template<class T>class Counter
2.  {
3.     static int counter;
4.     const int id;
5.  public:
6.     Counter():id(++counter){}
7.     int getId(){return id;}
8.     static int getCounter(){return counter;}
9.  };
10.
11. class B: public Counter<B> {};
12. class C: public Counter<C> {};
13.
14. // in Counter.cpp:
15. template<class T>
16. int Counter<T>::counter = 0;
17.
18. //----------------
19. //   main.cpp
20. //----------------
21. int main()
22. {
23.    B b;
24.    cout << B::getCounter() << endl << b.getId() << endl;
25.    cout << C::getCounter() << endl;
26.
27.    C* pc = new C();
28.    cout << B::getCounter() << endl;
29.    cout << C::getCounter() << endl << pc->getId() << endl;
30.
31.    return 1;
32. }
```

הערה:

הקוד שלמעלה עובר קומפילציה. השימוש ב-Template במחלקה Counter ובנורשים B ו-C נועד בכדי ליצור עותק ייחודי של המשתנה הסטטי counter ל-B ול-C בנפרד, תוצאה שמושגת כיון שכל אחד מהם יורש ממחלקה ממש אחרת.

ל-pattern הזה קוראים Curiously Recurring Template Pattern, או בקיצור CRTP.

שאלה 1

מה יודפס בשורות 24 ו-25? (=> מסמן ירידת שורה).

א. 0 => 1 => 0

ב. 1 => 0 => 0

ג. 1 => 1 => 0

ד. 0 => 0 => 1

ה. 2 => 1 => 1

ו. אף תשובה אינה נכונה.

שאלה 2

מה יודפס בשורות 28 ו-29? (=> מסמן ירידת שורה).

א. 1 => 1 => 1

ב. 2 => 1 => 1

ג. 2 => 2 => 2

ד. 1 => 2 => 2

ה. 2 => 2 => 1

ו. אף תשובה אינה נכונה.

שאלה 3

מורן שירן ולירן התכוננו למבחן ועברו על הקוד לעיל.

לירן טען: אפשר להחליף את שורה 27 בקוד הבא ללא שום שינוי בפלט של התוכנית:

```
Counter<C>* pc = new C();
```

שירן טען: ~~אפשר היה להפוך את הפונקציה~~ getId ~~בשורה 7 לפונקציה סטטית, כיון שהמשתנה~~ id ~~הוא ממילא~~ const.

מורן טען: שורות 15-16 מיותרות למעשה, מכיון שמשתנה סטטי ממילא נולד ומאותחל ל-0.

מי מהשלושה צודק?

א. לירן (הוא גם מצטיין דיקאן).

ב. לירן ומורן.

ג. שירן ומורן.

ד. ~~שלושתם צודקים.~~

ה. ~~שלושתם טועים.~~

ו. ~~אף אחת מהתשובות אינה נכונה.~~

שאלות 4-7 מתייחסות לקטע הקוד הבא.

```
1. class Three : public Two
2. {
3. public:
4.     Three(const Two& two): Two(two) {}
5.     Two& operator*(){return *this;}
6. };
7.
8. int main()
9. {
10.    One* pOne = new Two();
11.    Three& three = pOne->doIt();
12.    *three = *pOne;
13.    three = *pOne;
14.    return 1;
15.}
```

נתון שהקוד עובר קומפילציה בהצלחה.

שאלה 4

אילו מהטענות הבאות הכרחיות (מעבר לקיים בתוכנית) על-מנת ששורה 11 תעבור קומפילציה?

א. doIt חייבת להחזיר אובייקט מסוג Three.

ב. doIt חייבת להחזיר טיפוס שיכול להימסר ל-Three בהשמה.

ג. doIt חייבת להחזיר טיפוס שיכול להימסר לבנאי של Three.

ד. doIt חייבת להחזיר אובייקט ByRef מסוג Three או נגזריו.

ה. doIt חייבת להחזיר אובייקט ByRef מסוג Three או אב ישיר או עקיף שלו.

ו. אף תשובה אינה נכונה (אף טענה אינה הכרחית או ישנה יותר מטענה הכרחית אחת).

שאלה 5

אילו מהטענות הבאות הכרחיות (מעבר לקיים בתוכנית) על-מנת ששורה 12 תעבור קומפילציה?

א. המחלקה One בהכרח העמיסה אופרטור *.

ב. המחלקה Two בהכרח העמיסה אופרטור *.

ג. למחלקה Two יש בנאי שיכול לייצר אובייקט Two באמצעות פרמטר מסוג One ובנאי זה איננו explicit או לחילופין למחלקה Two יש אופרטור השמה שמצפה לקבל אובייקט One.

ד. במחלקה One יש אופרטור casting ל-Three או ל-Two.

ה. טענות ג' או ד' מספקות (אפשר להסתפק בכל אחת מהן).

ו. אף תשובה אינה נכונה.

שאלה 6

הניחו כעת כי נתון שלא נכתב מפורשות אופרטור השמה באף אחת מהמחלקות, וכן כי לא נכתב מפורשות באף אחת מהמחלקות בנאי שמקבל כפרמטר אובייקט מהמשפחה One, Two, Three – לשם כך גם שורה 4 הופכת להיות בהערה. כמו כן הנח שאף אחת מהמחלקות לא מימשה אופרטור casting. תחת ההנחות לעיל, אילו מהשורות לא תעבורנה קומפילציה?

א. כל השורות מ-10 עד 13.

ב. כל השורות מ-11 עד 13.

ג. שורות 12 ו-13 בלבד.

ד. שורה 13 בלבד.

ה. כל השורות יכולות לעבור קומפילציה.

ו. אף תשובה אינה נכונה.

שאלה 7

מורן שירן ולירן התכוננו למבחן ועברו על הקוד לעיל.

לירן טען: במחלקות One ו-Two בהכרח מימשו אופרטור *.

שירן טען: יתכן שהפונקציה doIt לא מומשה במחלקה One אלא רק במחלקה Two, ובלבד שהכרזנו עליה ב-One כפונקציה וירטואלית טהורה.

מורן טען: המחלקה Two בהכרח יורשת מהמחלקה One בהרשאת private.

מי מהשלושה צודק ?

א. לירן (הוא גם מצטיין דיקאן).

ב. שירן.

ג. מורן.

ד. שלושתם צודקים.

ה. שלושתם טועים.

ו. אף תשובה אינה נכונה.

חלק ב׳ – שאלת תכנות (51 נק׳)

חברת BetterParking מעוניינת לפתח מערכת משוכללת לניהול חניונים.

עליך להציג באופן מלא את המחלקות הנדרשות וכן לממש את הנושאים הבאים:

(א) קריאת הקונפיגורציה של החניון מקובץ.

(ב) מערכת לניהול משימות עבור עובדי החניון כולל ממשק משתמש לעובדי החניון.

(ג) איתור מקום חניה מתאים.

מערכת החניון פועלת באופן הבא:

לקוח שמגיע לחניון בוחר במערכת מאפייני חניה והמערכת מציעה לו מקום חניה המיועד עבורו. מערכת החניון יודעת להפנות את הנהג למקום החניה המתאים לפי צרכיו וכאשר הלקוח חוזר לקחת את רכבו המערכת יודעת להפנותו לפי מספר הרכב למיקום החניה בו נמצא הרכב.

שירותים / אילוצי החניה לדוגמה שלקוח יכול להעלות:

- שירות רחיצה
- שירות טעינה לרכב חשמלי
- שירות קישוט רכב חתן-כלה
- חניה מוצלת אילוץ מחייב לכל משך החניה
- רכב מונע בגז (יכול לחנות רק באיזורים מסויימים – אילוץ מחייב לכל משך החניה)
- רכב גבוה (יכול לחנות רק באיזורים מסויימים – אילוץ מחייב לכל משך החניה)

הלקוח יכול לבקש כל קומבינציה של אילוצים וכן כל קומבינציה של שירותים. כל האילוצים חייבים להתקיים, אולם רק שירות אחד יכול להתבצע בזמן נתון. כאשר השירות הראשון מסתיים המערכת תחפש מקום מתאים לפי השירות השני שנדרש ותייצר משימה להעביר את הרכב בהתאם. ההוראה תישמר במערכת עד שתילקח לביצוע ע״י אחד מעובדי החניון. באחריותך בין השאר לנהל את מערכת המשימות של עובדי החניון.

תמיכה במשימות לעובדי החניון:

כאשר עובד חניון מסיים הזזת רכב הוא יוכל להודיע זאת למערכת דרך הממשק. עובד החניון יכנס לתפריט עובדים, יזהה את עצמו לפי מספר עובד ויוצגו לו כל המשימות שהופנו אליו, העובד יוכל לצאת מהרשימה או לבחור לדווח שמשימה בוצעה (למשל באמצעות מספר רץ של המשימה). לאחר דיווח על ביצוע המשימה, ממשק המערכת ידע לבקש פרטים נוספים בהתאם למשימה שהסתיימה, למשל – עבור משימה של הזזת רכב, הממשק יבקש את המיקום החדש של הרכב. באחריותך לדאוג שהנתון הנ״ל יעודכן במערכת החניון.

סימון משימה כבוצעה, באמצעות הממשק, אמור לגרום להסרתה מרשימת המשימות הפתוחות ובאחריותך לנהל נושא זה.

מנוע המערכת מאפשר לממשק לקבל את רשימת כל המשימות הפתוחות (שעדיין לא נלקחו ע״י עובד כלשהו) – באמצעות Parking::getUnassignedTasks, וכן המשימות שבביצוע (שנמסרו לעובד – לפי מספר עובד: Parking::getTasksByWorker(long workerId).

עובד פנוי (שאין לו משימות כרגע) יכול לגשת לממשק מיוזמתו, לבקש את רשימת המשימות שטרם הופנו לעובד ולהפנותן דרך הממשק לעובד כלשהו לרבות לעצמו.

בסיום טעינה חשמלית של רכב, מערכת טעינת החשמל תייצר באופן אוטומטי משימה ותמסור אותה למערכת המשימות שלך. באחריותך לתמוך באפשרות זו ע״י הכנת פונקציה מתאימה שמערכת החשמל תוכל לקרוא לה. כמו כן, בסיום כל שירות ״ידני״ כמו רחיצה של רכב או קישוט

רכב חתן-כלה, עליך לאפשר לעובד החניון לדווח על סיום השירות באמצעות ממשק המשתמש (זו אינה פעולה אוטומטית) ולהפעיל פונקציה מתאימה שתדווח למנוע על סיום קבלת השירות. במידה ושירות שהרכב ממתין לו אמור להתחיל ע״י פעולה ״ידנית״ כגון חיבור להטענה, רחיצה, קישוט רכב וכו׳, עליך לאפשר יצירה של משימה מתאימה ולתמוך בה.

ככלל סביר שמשימה תכיל משתנה enum שיתאר מה עובד החניון נדרש לעשות וכן הפניה לרכב הרלבנטי. יתכן ויידרשו משתנים נוספים לשיקולך.

רצוי שהמימושים שנדרשים עבור כל נושא ניהול המשימות יהיו גנריים ויתאימו גם לשירותים אחרים בעתיד – אפשרי להשתמש באותה פונקציה לכל הצרכים. הנח שלמערכת טעינת החשמל ולממשק המשתמש ישנה גישה לאובייקט מערכת הניהול של החניון באמצעות הפונקציה הסטטית Parking::getParkingMngSys() שמחזירה reference אל מערכת ניהול החניון (אובייקט יחיד וקיים מסוג Parking).

<u>נתוני החניון</u>

- לחניון ישנם מספר מקטעי חניה, נשתמש במקטעים על-מנת להגדיר קומות שונות בחניון או איזורים בעלי מאפיינים שונים.
- למקטע בחניון יהיה <u>מספר קומה</u> <u>ואות כלשהי באנגלית</u>.
- כל מקטע בחניון מתאפיין באוסף המאפיינים הספציפיים שלו, כלומר – <u>לאיזה אילוצים הוא עונה ואילו שירותים הוא יכול לתת</u>: האם מותר להכניס למקטע זה רכבים מונעי גז, האם המקטע מוצל או פתוח לשמש, האם זהו מקטע רחיצת מכוניות וכו׳. מומלץ להחזיק את האילוצים והשירותים שמאפיינים מקטע באמצעות שני משתנים מסוג long – אחד לשירותים ואחד לאילוצים – לניהול מאפייני מקטע החניון <u>באמצעות דגלים ביטיים</u>.
  גם את דרישות הלקוח כדאי לנהל באמצעות שני משתנים מסוג long.
- <u>מקטע חניון מכיל כמות כלשהי של חניות</u>, כאשר לכל חניה יש <u>מספר סידורי</u>, ובנוסף – <u>פוינטר לרכב שכרגע חונה בחניה זו</u>, אם ישנו.
- מעוניינים שניתן יהיה במהירות ובקלות להגיע <u>ממספר רכב למיקומו בחניון</u> בלי צורך לסרוק את כל החניון.
- בשורה משמחת: בחניון שלנו <u>אין תעריפי חניה ואין תשלום על שירותים</u>! (הכל <u>בחינם</u>).

<u>שים לב</u>:
יש לענות על סעיפי השאלה <u>באופן מסודר לפי הסעיפים בעמוד הבא</u>.

סעיף א׳ (25 נקודות)

כתוב את הגדרת המחלקות הנדרשות לפתרון (כל ה-prototypes, ללא מימושים).
יש להציג את כל המחלקות הנדרשות לניהול החניון ומשימות לעובדי החניון לרבות כל החתימות של הפונקציות (למעט getters ו-setters), גם אלו שאינך נדרש לממש ובוודאי אלו שעליך לממש.

הקפד על שימוש נכון ב-public, protected, private ו-const.

סעיף ב׳ (12 נקודות)

ממש באופן מלא את כל הפונקציות הנדרשות לקריאת נתוני מערכת החניון מקובץ.

יש לטפל בקריאת מקטעי החניון לפי התיאור המדוייק שבשאלה. הנך חופשי להחליט לגבי מבנה הקובץ וכן האם להשתמש בקובץ טקסט או קובץ בינארי.

סעיף ג׳ (10 נקודות)

ממש באופן מלא את כל הפונקציות הנדרשות לניהול משימות לעובדי החניון, כולל ממשק המשתמש הנדרש.

סעיף ד׳ (4 נקודות)

ממש את הפונקציות הנדרשות לאיתור מקום חניה מתאים ללקוח חדש.

הצעה לפתרון:

1. רוץ על כל מקטעי החניה, שאל כל מקטע:
ParkingSpot* ParkingSection::getAvailableMatch(long requirements, long services)
2. הבדיקה בתוך מקטע החניה תתבסס על שני long-ים, אחד לאילוצים ואחד לשירותים, שמוחזקים בתוך מקטע החניה:

(א) על-מנת לוודא שכל האילוצים מתקיימים, נדרשת בדיקת bitwise בין שני long באופן הבא: מבצעים & בין שניהם ואם התוצאה שווה לדרישות הלקוח האילוצים מתקיימים!

(ב) אם הלקוח ביקש שירותים כלשהם, לפחות שירות אחד חייב להתקיים. נדרש קודם כל לבדוק אם הלקוח ביקש בכלל שירותים (ה-long שונה מאפס), אם כן, נדרשת בדיקת bitwise בין שני long באופן הבא: מבצעים & בין שניהם ואם התוצאה שונה מאפס סימן שלפחות שירות אחד מתקיים – אין צורך לדעת מהו, יטופל ע״י המערכת בקוד שלא באחריותך לאחר שהרכב יגיע למקום החניה.

סוף !