מספר זהות:

סמסטר א' תשע"ב
מועד: א' 01/02/2013
שעה: 9:00
משך הבחינה: 3½ שעות
חומר עזר: כל חומר עזר כתוב מותר

בחינה בקורס: **תכנות מכוון עצמים ושפת C++**

מרצים: **אמיר קירש, ד"ר איריס רוזנבלום**

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 כתוב 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 1-7 מתייחסות לקטע הקוד הבא:

```
1.  template<class ELEMENT> class Array
2.  {
3.     class Element
4.     {
5.        Array<ELEMENT>* pArray;
6.        int index;
7.     public:
8.        Element(Array<ELEMENT>* p, int i)
9.           : pArray(p), index(i) {}
10.       const Element& operator=(const ELEMENT& e) {
11.          pArray->set(index, e); // call copy-on-write
12.          return *this;
13.       }
14.       operator ELEMENT()const {
15.          return pArray->arr[index];
16.       }
17.    };
18.
19.    friend class Element;
20.    ELEMENT* arr;
21.    int size;
22.    int* ref_counter;
23.    void attach(const Array& a) {
24.       arr = a.arr; size = a.size;
25.       ref_counter = a.ref_counter;
26.       ++(*ref_counter);
27.    }
28.    void detach() {
29.       if(--(*ref_counter) == 0) {
30.          delete []arr;
31.          delete ref_counter;
32.       }
33.    }
34.    void set(int index, const ELEMENT& e) {
35.       if(*ref_counter > 1) { // need copy-on-write!
36.          Array temp = clone();
37.          detach();
38.          attach(temp);
39.       }
40.       arr[index] = e;
41.    }
42. public:
43.    explicit Array(int);
```

```
44.     Array<ELEMENT> clone()const;
45.     Array(const Array<ELEMENT>& a){attach(a);}
46.     ~Array(){detach();}
47.     const Array& operator=(const Array<ELEMENT>& a) {
48.        detach(); attach(a); return *this;
49.     }
50.     Element operator[](int index) {
51.        return Element(this, index);
52.     }
53.     const ELEMENT& operator[](int index)const {
54.        return arr[index];
55.     }
56. };
57.
58. template<class ELEMENT>
59. Array<ELEMENT>::Array(int size1)
60.     : size(size1), ref_counter(new int(1))
61. {
62.     arr = new ELEMENT[size];
63. }
64.
65. template<class ELEMENT>
66. Array<ELEMENT> Array<ELEMENT>::clone()const {
67.     Array temp(size);
68.     for(int i=0; i<size; ++i) {
69.        temp.arr[i] = arr[i];
70.     }
71.     return temp;
72. }
73.
74. int main()
75. {
76.     Array<int> a1(1), a2(2);
77.     Array<int>* ptr3 = new Array<int>(3);
78.     a2[0] = 1;
79.     a2[1] = 2;
80.     a1 = a2;
81.     (*ptr3)[0] = a1[0] + a2[1];
82.     (*ptr3)[1] = a1[1] + a2[0];
83.     cout << (*ptr3)[0] << ", " << (*ptr3)[1] << endl;
84.     delete ptr3;
85.     return 1;
86. }
```

נתון שהתוכנית לעיל עוברת קומפילציה בהצלחה!

שאלה 1

כמה פעמים התוכנית עוברת בשורה 62?

א. פעם אחת.

ב. פעמיים.

ג. שלוש פעמים.

ד. ארבע פעמים.

ה. חמש פעמים!

ו. אף תשובה אינה נכונה.

שאלה 2

כמה פעמים התוכנית נכנסת לפונקציה set שבשורה 34?

א. שלוש פעמים.

ב. ארבע פעמים.

ג. שש פעמים.

ד. שמונה פעמים.

ה. עשר פעמים.

ו. אף תשובה אינה נכונה.

שאלה 3

כמה פעמים התוכנית עוברת בשורה 36?

א. אפס פעמים.

ב. פעם אחת.

ג. פעמיים.

ד. שלוש פעמים.

ה. זה לא ידוע, מכיון שאנחנו לא יודעים כיצד מבוצעות ההשמות של שורות 78 ו-79.

ו. אף תשובה אינה נכונה.

שאלה 4

מה יודפס בשורה 83?

א. 0, 0

ב. 1, 2

ג. 2, 4

ד. 3, 3

ה. יודפסו שני מספרים זבליים כיון של-a1[1] אין איתחול.

ו. אף תשובה אינה נכונה.

שאלה 5

כמה פעמים נעבור ב-d'tor של המחלקה Array עד שורה 84 כולל?

(השאלה אינה כמה פעמים יבוצע delete על הקצאה, אלא על עצם המעבר ב-d'tor שבשורה 46).

א. פעם אחת.

ב. פעמיים.

ג. שלוש פעמים.

ד. ארבע פעמים.

ה. חמש פעמים.

ו. אף תשובה אינה נכונה.

שאלה 6

כמה פעמים נעבור ב-d'tor של המחלקה Array בעקבות שורה 86?

(השאלה אינה כמה פעמים יבוצע delete על הקצאה, אלא על עצם המעבר ב-d'tor שבשורה 46).

א. אפס פעמים.

ב. פעם אחת.

ג. פעמיים.

ד. שלוש פעמים.

ה. ארבע פעמים.

ו. אף תשובה אינה נכונה.

שאלה 7

מורן שירן ולירן התכוננו למבחן ועברו על הקוד לעיל.

לירן טען: ב-main שמופיע במבחן לא מבוצע בכלל copy on write.

שירן טען: ב-main שבמבחן אנחנו לעולם לא עוברים באופרטור [] ה-const-י (של שורה 53) כיון שבמהלך התוכנית לא נוצר או מתקבל בפונקציה אובייקט Array כ-const שמנסה להפעיל אופרטור [].

מורן טען: לא ניתן בתוכנית זו לייצר מערך באמצעות התחביר הבא:

```
Array<int> a1 = 1;
```

מי מהשלושה צודק ?

א. לירן (הוא גם מצטיין דיקאן).

ב. לירן ומורן.

ג. לירן ושירן.

ד. שלושתם צודקים.

ה. שלושתם טועים.

ו. אף אחת מהתשובות אינה נכונה.

חלק ב׳ – שאלת תכנות (51 נק׳)

חברת iRovac מפתחת שואבי אבק רובוטיים אשר יודעים באופן אוטומטי לצאת מבסיס ההטענה שלהם, לנקות את חדרי הבית, לחזור להטענה כאשר רמת ההטענה יורדת לרמה נמוכה ולסיים את עבודת הניקוי כאשר האלגוריתם של הרובוט מזהה שכל הנקודות בבית נוקו לפי הכללים.

מחלקת פיתוח האלגוריתם של הרובוט מעוניינת לבחון גירסאות שונות של האלגוריתם ולכן הוחלט לכתוב סימולטור – שאותו עליך לממש – להשוואה בין אלגוריתמים של רובוטים.

הסימולטור יטען נתונים של אוסף בתים וכן מספר כלשהו של מימושי אלגוריתם, יריץ את האלגוריתמים השונים על הבתים השונים, יאסוף את התוצאות וידווח על ציוני האלגוריתמים.

ייצוג בית

בית מאופיין ע״י קובץ טקסט בפורמט הבא: בשורה הראשונה יופיע שם הבית (מחרוזת ללא רווחים), בשורה השניה מספר המציין את רוחב הבית מבחינת כמות התווים המייצגים אותו), בשורה השלישית אורך הבית (כלומר, מספר השורות), מייד לאחר מכן שורות המייצגות את הבית כאשר התו W מציין קיר (או חפץ כלשהו שלא ניתן לעבור דרכו), התו D מציין את תחנת העגינה של הרובוט שממנה הוא יוצא לדרכו בראשונה ואליה הוא חוזר לטעינה, R מציין שטיח ורווח או כל תו אחר מסמן רצפה רגילה. כאשר בשורה ישנם יותר תווים מרוחב הבית שצויין יש להתעלם מהתווים העודפים. כאשר בשורה חסרים תווים ניתן להניח שכל התווים החסרים הם קירות (התו W) – כנ״ל לגבי שורות עודפות או חסרות.

קובץ שחסר בו התו D או שמופיע יותר מפעם אחת – ייזרק Exception מתאים בנקודה שבה הבעיה מתגלית, ה-Exception ייתפס בתוכנית במקום המתאים והתוכנית תוציא הודעה שאנו מתעלמים מקובץ קלט זה, כולל תיאור הבעיה, אך תמשיך לקבצים האחרים.

ייצוג אלגוריתם

אלגוריתם של רובוט מיוצג ע״י קובץ מקומפל שמכיל שני דברים:

(א) מימוש של מחלקה היורשת מהמחלקה האבסטרקטית RobotAlgorithm וכוללת מימוש של המתודה הבאה שהוכרזה כאבסטרקטית במחלקת הבסיס:

```
virtual int move();
```

הערך המוחזר של פונקציה זו הינה הכיוון שהרובוט בוחר לנוע בו:
0 = מעלה, 1 = מטה, 2 = ימינה, 3 = שמאלה.

(ב) שורת יצירה של אובייקט גלובלי מסוג AlgorithmRegister באופן הבא:

```
AlgorithmRegister algorithm1("Algorithm1", new Algorithm1());
```

כאשר, Algorithm1 יוחלף בשם המחלקה שיצרנו ב-(א) ושאותה אנו רוצים לבחון.

ישנה פונקציית מערכת המאפשרת לטעון קובץ מקומפל בזמן ריצה, יש לעשות שימוש בפונקציה זו על-מנת לטעון את האלגוריתמים, הפרוטוטייפ של הפונקציה הינו:

```
int loadLibrary(const char* lib);
```

כאשר, lib הוא שם הקובץ המקומפל, אם לא נמסר path אזי ההנחה שהקובץ לטעינה נמצא בספריה הנוכחית. הערך המוחזר הינו אפס אם הטעינה הצליחה וקוד שגיאה כלשהו אם לא.

המשתמש הולך להפעיל את הסימולטור באמצעות ה-command line באופן הבא:

iRovac houses: <house-file1> <house-file2> <house-file3> <...> algo: <algo-file1> <algo-file2> <algo-file3> <...>

יש למסור לפחות שם קובץ לבית אחד ולפחות שם קובץ לאלגוריתם אחד.

טעינת קבצי האלגוריתם אמורה לרשום אותם אוטומטית לתוך מבנה נתונים סטטי שעליך ליצור בתוך המחלקה AlgorithmRegister שעליך לממש, כולל בנאי מתאים שיבצע בפועל את הרישום של כל האלגוריתמים שיש להריץ.

כאשר התוכנית מסיימת לטעון את כל קבצי הבתים וקבצי האלגוריתם, יש להריץ את כל האלגוריתמים על כל הבתים ובסוף התהליך להדפיס למסך את ריכוז הניקוד של כל האלגוריתמים, בפורמט הבא (אין צורך להציג טבלה, אבל יש להדפיס את הנתונים עצמם כך):

	algo1-filename	algo2-filename	algo3-filename	...
house1-name	<grade>	<grade>	<grade>	<grade>
house2-name	<grade>	<grade>	<grade>	<grade>
house3-name	<grade>	<grade>	<grade>	<grade>
...				
	<TOTAL>	<TOTAL>	<TOTAL>	<TOTAL>

הרצת האלגוריתמים וחישוב הציון:

1. עליך לדאוג במחלקה RobotAlgorithm שבאחריותך, לממש את המתודה (הלא וירטואלית):
```
void init(Sensor* pSensor)
```
מתודה זו אמורה לאתחל את האלגוריתם ולמסור לו פוינטר לסנסור שבאמצעותו הרובוט יוכל לשאול שאלות כמו מהו סוג המשבצת הבאה משמאל, מימין, למעלה או למטה, ע״י:
```
pSensor->getCellType(int direction)
```
2. בתחילת הריצה של כל אלגוריתם, המחלקה המנהלת שלך תאתחל את האלגוריתם עם פוינטר לבית שעליו רצים כרגע ע״י קריאה למתודה init שצויינה למעלה. הבית יירש מהמחלקה סנסור, כך שהרובוט לא באמת יכיר את הבית אלא רק יוכל לברר מהו הסוג של 4 המשבצות הצמודות.
3. בכל ״תור״ יש להורות לאלגוריתם לנוע (move) ולעקוב אחר התנועה שלו. הרובוט אמור להיות מספיק חכם ולא לנסות להיכנס לתוך קירות, אבל אם בכל זאת הוא מנסה, עליך לאסור זאת ולהשאיר את הרובוט במשבצת שבה הוא נמצא.
4. שואב האבק הרובוטי יודע להבחין בין משטח רצפה רגיל לבין שטיח. לפי הכללים עליו לעבור לפחות פעם אחת על משטחי רצפה רגילים ולפחות פעמיים (כלומר שהייה של שני תורות, רצופים או שאינם רצופים) על כל נקודה בשטיחים. כאשר הרובוט עבר על משטח רצפה פעם אחת יש לסמן אותה כנקיה, כאשר הרובוט עבר על משבצת שטיח פעם אחת המשבצת תהיה חצי נקיה ורק בביקור השני (שיכול להיות בתור העוקב, אך לא חייב) ניתן לסמן את משבצת השטיח כנקיה.
5. הסוללה של הרובוט מחזיקה 300 תורות (קריאות ל-move). המתנה של תור אחד בעמדת הטעינה טוענת את הרובוט ב-1/10 כלומר בכוח ל-30 תורות, כך שלצורך טעינה מלאה נדרשת המתנה של 10 תורות בעמדת הטעינה, אך האלגוריתם אינו חייב להמתין לטעינה מלאה. הסימולטור צריך לעקוב אחר מצב הסוללה, במידה והסוללה התרוקנה לגמרי והרובוט אינו בעמדת הטעינה הוא ידום בנקודה שבה עצר ולא ימשיך.
6. על הסימולטור להריץ את כל האלגוריתמים במקביל על כל בית, ברגע שאלגוריתם אחד סיים לנקות את הבית באופן מלא ושב לנקודת הטעינה יש לבצע עוד 100 תורות (קריאות ל-move) ואז לסיים את הסימולציה על בית זה ולעבור לבית הבא. כמו כן, במידה ואף אלגוריתם לא סיים את ניקיון הבית במשך 9,000 תורות יש לעצור את הסימולציה על בית זה ולעבור לבית הבא.
7. הניקוד של כל אלגוריתם עבור כל בית, יחושב באופן פשוט לפי מספר המשבצות הנקיות בתום הריצה על הבית (יכול להיות בחצאים, בגלל ניקוי חלקי של משבצות שטיח).

שים לב:

הקפד שלא לכתוב ממשק משתמש אלא רק בפונקציות המיועדות לכך!

סעיף א׳ (21 נקודות)

כתוב את הגדרת כל המחלקות הנדרשות לפתרון (כל ה-prototypes, ללא מימושים).

הצג גם הגדרה של מחלקת אלגוריתם אחת לדוגמה.

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

סעיף ב׳ (20 נקודות)

ממש באופן מלא את כל הפונקציות הנדרשות להרצת סימולציה והצגת תוצאותיה, כולל main.

סעיף ג׳ (10 נקודות)

ממש אלגוריתם כלשהו לרובוט.

- אלגוריתם נאיבי שמזיז את הרובוט באופן כלשהו ונמנע מלהיתקל בקירות, יזכה בחמש נקודות.
- אלגוריתם שבנוסף לאמור למעלה מעדיף לנקות משבצות חדשות מאשר כאלו שהוא כבר ניקה, יזכה ב-10 נקודות. שימו לב שהמידע אילו נקודות כבר נוקו צריך להישמר ברובוט, אסור לרובוט והוא גם אינו יכול, להיעזר במחלקת הבית על מנת להכיר את מבנה הבית או לשאול מה עדיין לא נקי מכיון שהאלגוריתם בעולם האמיתי לא יוכל לשאול את הבית שאלות כאלו. לצורך זכירת נתונים לגבי משבצות נדרש להיעזר במבנה נתונים מסוג map<Point, bool> כאשר אין צורך לממש את המחלקה Point, ניתן להניח שהיא קיימת ומומשה כיאות, מבנה הנתונים יזכור באילו נקודות כבר ביקרנו כאשר נקודה מיוצגת באמצעות קואורדינטות X ו-Y.
- אלגוריתם שבנוסף ללמעלה יודע גם להתייחס באופן נכון למשבצות שטיח, יזכה ב-13 נקודות (כלומר, 3 נקודות בונוס). לצורך זכירת הנתונים לגבי משבצות נדרש להיעזר כעת במבנה נתונים משוכלל יותר, למשל: map<Point, CellInfo>, אין צורך לממש את המחלקה Point, ניתן להניח שהיא קיימת ומומשה כיאות, אבל את המחלקה CellInfo יש לממש.
- אלגוריתם שעושה את מה שכתוב למעלה וגם יודע לחזור לעמדת הטעינה בזמן כדי לא להיתקע, יזכה ב-16 נקודות (כלומר, 6 נקודות בונוס). מותר לממש את החזרה לעמדת הטעינה ע"י חזרה מדוייקת לאחור על כל הצעדים שעשינו מאז היציאה מהטעינה.

סוף !