060060 4440

המסלול האקדמי המכללה למינהל

ביה"ס למדעי המחשב

המסלול האקדמי
המכללה למינהל

ברקוד נבחן

ת.ז הסטודנט: _____

מספר חדר: _____

מספר נבחן: _____

מספר אסמכתא: _____

מבחן בקורס: תכנות מונחה עצמים

תאריך הבחינה: 19.08.15

שנת הלימודים: תשע"ה, **סמסטר:** ב', **מועד:** ב'

משך הבחינה: 3 שעות

שם המרצה/ים:	שם המתרגל/ים:
אליהו חלסצ'י	אילן לופו
איגור רוכלין	חיים שפיר

מבנה הבחינה: הבחינה מורכבת מחלק אחד.

מספר השאלות הכולל בבחינה: 4.

משקל כל שאלה: בצמוד לכל שאלה

הוראות לנבחן:

- אסור השימוש בכל חומר עזר
- יש לענות בגוף השאלון.
- נדרש להחזיר את השאלון.
- לא מצורף נספח לבחינה
- מחברת טיוטה: כן
- מחברת נפרדת לכל שאלה: לא

בהצלחה!!

שאלה 1 – ירושה והכלה (25 נק')

הגדרת מחלקה מורכבת משם המחלקה, מה היא יורשת, וה data members שלה. לדוגמא:

```
class B : public A{
        int x,y;
```

** **הערה עבור כל המבחן**: עבור טיפוסים שלא הוגדר עבורם מה עליהם להכיל, אין צורך ליצור עבורם data members, או מתודות עזר.

הגדירו את המחלקות המייצגות את האלמנטים הבאים:

- רובוט קרקעי (Ground Robot)
  - לרובוט יש מצלמה, ספק כוח חשמלי, זרוע מכנית, וארבעה גלגלים.
- רובוט רחפן (Quadcopter Robot)
  - לרובוט יש מצלמה, ספק כוח חשמלי, זרוע מכנית, וארבעה רוטורים.
- רובוטריק (Transformer Robot) הוא גם רובוט קרקעי וגם רחפן.
  - לרובוט יש מצלמה, ספק כוח חשמלי, זרוע מכנית, ארבעה גלגלים, וארבעה רוטורים.

תשובה + מפתח בדיקה:

- טיפוסים עבור המצלמה, ספק הכוח, הזרוע המכנית, גלגל, ורוטור – 5 נק'
- הגדרת טיפוס רובוט כללי, המכיל משתנים המשותפים לכל הרובוטים – 5 נק'
- הגדרת הרובוט הקרקעי
  - כיורש וירטואלי של הרובוט הכללי – 3 נק'
  - הכלה של ארבעה גלגלים (מערך של [4] או *Wheel) – 2 נק'
- הגדרת הרחפן
  - כיורש וירטואלי של הרובוט הכללי – 3 נק'
  - הכלה של ארבעה רוטורים (מערך של [4] או *Rotor) – 2 נק'
- הגדרת הרובוטריק כיורש של הרובוט הקרקעי והרחפן – 5 נק'.

```
class Camera{};
class ElectricPowerSupplier{};
class MechanicalArm{};
class Wheel{};
class Rotor{};

class Robot{
protected:
        Camera c;
        ElectricPowerSupplier eps;
        MechanicalArm arm;
};
class GroundRobot: public virtual Robot{
protected:
        Wheel wheels[4];
};
class QuadcopterRobot : public virtual Robot{
protected:
        Rotor rotors[4];
};

class TransformerRobot : public GroundRobot, public QuadcopterRobot{};
```

שאלה 2 – constructors / destructors (15 נק')

כתבו בנאי מונחה עצמים עבור הרובוטריק, ממשו (רק) בנאים נוספים ע"פ הצורך.

תשובה:

הבנאי של הרובוטריק צריך לקבל מופע של רובוט קרקעי ומופע של רחפן (2 נק'), הפרמטרים צרכים להתקבל by ref ומוגנים כ const (1 נק'). באמצעותם עליו לאתחל בשורת האתחול (1 נק') את החלק הרובוטי שלו (2 נק'), החלק הקרקעי שלו (1 נק') , ואת החלק הרחפני שלו (1 נק')

```
TransformerRobot(const GroundRobot& gr, const QuadcopterRobot&
qr):Robot(gr),GroundRobot(gr),QuadcopterRobot(qr){}
```

כדי שקוד זה יעבוד צריך לבנות copy constructors לרובוט, לרובוט הקרקעי, ולרחפן:

```
Robot(const Robot& r){
        c = r.c;
        eps = r.eps;
        arm = r.arm;
}

GroundRobot(const GroundRobot& gr) :Robot(gr){
        for (int i = 0; i < 4; i++)
                wheels[i] = gr.wheels[i];
}

QuadcopterRobot(const QuadcopterRobot& qr):Robot(qr){
        for (int i = 0; i < 4; i++)
                rotors[i] = qr.rotors[i];
}
```

- ה CCTORs מקבלים פרמטר מאותו הסוג by ref מוגנים כ const (3 נק')
- הבנאים של הרובוט הקרקעי והרחפן דואגים לאתחול Robot בשורת אתחול (2 נק')
- העתקה נכונה של מערך הגלגלים \ רוטורים (2 נק')

שאלה 3 – קבצים, פולימורפיזם (28 נק')

כתבו את הפונקציה saveAll המקבלת אובייקט מסוג ofstream, מערך של רובוטים, ואורך המערך. במערך זה יתכנו רובוטים מכל הסוגים. באמצעות אובייקט ה ofstream, על הפונקציה לשמור את כל הרובוטים בצורה בינארית בקובץ. עליה לעשות זאת באופן כזה שיהיה ניתן מאוחר יותר לטעון אותם.

ממשו מתודות עזר במחלקות השונות במידת הצורך.

<u>טיפ: שימו לב לא לשמור מידע מיותר.</u>

תשובה + מפתח בדיקה

הפונקציה saveAll צריכה לקבל מערך של מצביעים לרובוטים (**Robot) כדי שכל תא יוכל להצביע לכל סוג רובוט (2 נק').

תחילה עלינו לשמור את אורך המערך, כדי שנדע בעתיד כמה איברים לטעון מהקובץ (3 נק')

עבור כל אחד מהאיברים נצטרך להפעיל פעולת שמירה לטיפוס (1 נק') ופעולת שמירה לאיבר (1 נק')

```
void saveAll(ofstream& out, Robot** robots,int size){
        // save the size of elements
        out.write((char*)&size, sizeof(size));
        // save the types of the robots, and the robots
        for (int i = 0; i < size; i++){
                robots[i]->saveType(out);
                robots[i]->save(out);
        }
}
```

כדי שקוד זה יעבוד עלינו לממש נכון את saveType (5 נק') כך שכל איבר ישמור תחילה את טיפוסו. ניתן לעשות זאת באמצעות מתודה וירטואלית במחלקה Robot ולממש אותה שוב בכל המחלקות. בפחות קוד ניתן לתת מימוש אחד במחלקה Robot שעושה שימוש ב typeid ושומר למשל את שתי האותיות הראשונות של הטיפוס:

```
// in Robot class
void saveType(ofstream& out){
        char type[2];
        type[0] = typeid(*this).name[6];
        type[1] = typeid(*this).name[7];
        out.write(type, 2);
}
```

יש לשים לב שהפעלנו את typeid על this* ושדילגנו על " class" בשם שחזר (2/5 נק')

כמו כן, נצטרך ליצור מתודה וירטואלית (2 נק') במחלקה Robot ששומרת את המידע שמוגדר במחלקה זו (2 נק'). ניתן להניח שקיימות מתודות save בטיפוסים שלא הגדרנו להם את המידע שלהם.

```
virtual void save(ofstream& out){
        c.save(out);
        eps.save(out);
        arm.save(out);
}
```

כעת עלינו לממש מחדש (לדרוס) את save במחלקות GroundRobot ו QuadcopterRobot. במחלקות אלה נרצה להפעיל תחילה את מתודת ה save של Robot (2 נק') ואז להפעיל את השמירה של שאר האיברים (גלגלים \ רוטורים) (2 נק'). כפי שנגלה בהמשך, זה יכול להיות רעיון טוב להפריד בין שתי השמירות הנ"ל.

```
        // in GroundRobot class
protected:
        void saveWheels(ofstream& out){
                for (int i = 0; i < 4; i++)
                        wheels[i].save(out);
        }
public:
        virtual void save(ofstream& out){
                // save the robot's parts
                Robot::save(out);
                // save the wheels:
                saveWheels(out);
        }


        // in QuadcopterRobot class
protected:
        void saveRotors(ofstream& out){
                for (int i = 0; i < 4; i++)
                        rotors[i].save(out);
        }
public:
        virtual void save(ofstream& out){
                // save the robot's parts
                Robot::save(out);
                // save the rotors:
                saveRotors(out);
        }
```

במחלקת הרובוטריק חובה עלינו לממש את המתודה save (3 נק') אחרת תהיה לנו שגיאת דו-משמעות. נשים לב שהפעלה עיוורת של מתודת ה save מהמחלקה GroundRobot ואז מהמחלקה QuadcopterRobot תשמור לנו פעמיים את נתוני ה Robot... לכן היה זה רעיון טוב להפריד בין שתי פעולות השמירה ונוכל לדאוג שלא נשמור דבר מיותר. לדוגמא, נפעיל את מתודת ה save של GroundRobot (שבתורה גם שומרת את נתוני Robot) ואז נפעיל את מתודת שמירת הרוטורים (שירשנו מ QuadcopterRobot) בלבד.

```
        // in TransformerRobot
        virtual void save(ofstream& out){
                GroundRobot::save(out);
                QuadcopterRobot::saveRotors(out);
        }
```

שמירת כל המידע, ללא מידע מיותר (5 נק')

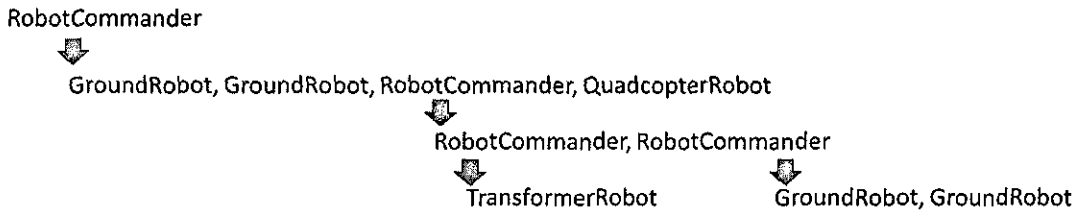
שמירת מידע כפול (0/5 נק')

שמירת חלק מהמידע (2/5 נק')

שאלה 4 – פולימורפיזם + generic algorithm (32 נק')

א. (6 נק') הגדירו מחלקה (ע"י שם, ירושה, ו data members בלבד) בשם RobotCommander עבור רובוט קרקעי שמהווה מפקד למערך של רובוטים. פקודיו יכולים להיות כל סוג של רובוט, ובפרט גם מסוג RobotCommander המפקד על מערך רובוטים משלו.

כלומר, אם נרצה, נוכל ליצור היררכיה פיקודית של רובוטים, כמו בתרשים:



ב. עליכם לממש את הפונקציות הבאות תוך כדי מינימום קוד כפול. ממשו מתודות עזר מונחות עצמים במחלקות השונות ע"פ הצורך ו\או template function כעזר.

1. בהינתן מופע של RobotCommander, הפונקציה getAirForce תחשב כמה רחפנים נמצאים תחת פיקודו (באופן עמוק). עבור הדוגמא לעיל התשובה היא 2 (הרי רובוטריק הוא גם רחפן).

2. בהינתן מופע של RobotCommander, הפונקציה getGroundForce תחשב כמה רובוטים קרקעיים נמצאים תחת פיקודו (באופן עמוק). עבור הדוגמא לעיל התשובה היא 8 (הרי גם מפקד וגם רובוטריק מהווים רובוטים קרקעיים).

תשובה + מפתח בדיקה

א. מחלקת מפקד שתירש את GroundRobot (3 נק') ותכיל מערך מסוג Robot** (3 נק') תספיק לנו. הרי GroundRobot הוא סוג של Robot, ולכן מערך זה יכול להכיל גם מפקדים.

```
class RobotCommander :public GroundRobot{
       Robot** robots;
};
```

ב. מה שמשותף לשני הסעיפים זה לעבור בצורה עמוקה על פני כל הרובוטים. מה ששונה זו השאילתה שעלינו לבצע. לכן נרצה לקבל אותה מבחוץ כפרמטר. שימוש ב object function יתאים לנו לצורך משימה זו.

נתחיל במימוש מתודת עזר שמקבלת object function כפרמטר ומבצעת את פעולתו על כל אברי המערך, ואם האובייקט הנוכחי הוא מפקד, אז נעפיל את אותה מתודת העזר מתוך אובייקט זה. כך נעמיק באופן רקורסיבי את הפעלת ה object function על כל הפקודים:

```
class RobotCommander :public GroundRobot{
	Robot** robots;
	int size;
public:
	template<class func>
	void applyToAllRobots(func& f){
		for (int i = 0; i < size; i++){
			f(robots[i]);
			if (typeid(*robots[i]) == typeid(RobotCommander)){
				RobotCommander* rc = (RobotCommander*)robots[i];
				rc->applyToAllRobots(f);
			}
		}
	}
};
```

- קבלת object function כפרמטר (3 נק')
- הפעלת ה object function על כל האיברים - כולל איברים מסוג מפקד (3 נק')
- שאילתה נכונה האם האיבר הנוכחי הוא מפקד, תשומת לב ל *, (3 נק')
- המרה נכונה של האיבר לטיפוס מצביע למפקד והפעלה (רקורסיבית) של מתודת העזר מתוכו (3 נק' המרה, 3 נק' הפעלה)

כעת נממש את שני ה object functions עבור ספירת הרחפנים וסיפרת הקרקעיים. נשים לב שעלינו להשתמש ב dynamic_cast מכיוון שנרצה לתפוס גם את המפקד כרובוט קרקעי, וגם את הרובוטריק כרובוט קרקעי וכרחפן. שימוש ב typeid תופס רק טיפוס אחד בלבד, ונצטרך יותר שאילתות...

```
struct IsAir{
	int count;
	IsAir(){
		count = 0;
	}
	void operator()(Robot* r){
		QuadcopterRobot* qr = dynamic_cast<QuadcopterRobot*>(r);
		if (qr)
			count++;
	}
};
struct IsGround{
	int count;
	IsGround(){
		count = 0;
	}
	void operator()(Robot* r){
		GroundRobot* gr = dynamic_cast<GroundRobot*>(r);
		if (gr)
			count++;
	}
};
```

שימוש בטיפוסים עבור ה object function (1 נק')

שימוש נכון ב dynamic_cast (5 נק')

שימוש עבור כל האופציות ב typeid (יותר קוד) (4/5 נק')

תפיסה של רק חלק מהטיפוסים (2/5 נק')

התאמה לטיפוס שנשלח במתודת העזר (*Robot בתשובה כאן) (2 נק')

ספירה נכונה (1 נק')

כעת נותר לנו לממש את שתי הפונקציות שנתבקשנו:

```
int getAirForce(RobotCommander& r){
	IsAir a;
	r.applyToAllRobots(a);
	return a.count;
}
int getGroundForce(RobotCommander& r){
	IsGround a;
	r.applyToAllRobots(a);
	return a.count;
}
```

מימוש נכון של הפונקציות ע"פ דרישות השאלה (2 נק')

בהצלחה