פקולטה: מדעי הטבע

מחלקה: מדעי המחשב

שם הקורס: מבנה זכרון ושפת C++
קוד הקורס: 7027810
תאריך בחינה: 20/7/2017
מועד: א
סמסטר: ב
משך הבחינה: שעתיים
שם המרצים: ד"ר אופיר פלא, ד"ר מירי בן ניסן
חומר עזר: פתוח
שימוש במחשבון: לא

<u>הוראות כלליות:</u>

- הניסוח הוא בלשון זכר מטעמי נוחות ומתייחס לכולם/ן!
- אין בחירה במבחן. יש לענות על כל השאלות.
- ניתן להסתמך על כל סעיף במבחן גם אם לא פתרתם אותו על מנת לפתור סעיף אחר במבחן.
- ניתן לרשום "לא יודע/ת" על סעיף ולזכות ב 20% מהנקודות המוקנות לסעיף הספציפי.
- במידה ולסעיף ניתנה תשובה ובנוסף נרשם לגבי הסעיף "לא יודע/ת" אזי הניקוד שיינתן לסעיף יהיה 0 מבלי שהתשובה תיקרא.
- אם לא רשום דבר בסעיף או שזה כלל לא נמצא ההנחה היא שנרשם "לא יודע/ת" עבור אותו סעיף.
- יורדו נקודות על פתרון נכון שאינו אופטימלי בהתייחס לנלמד בקורס.
- לכל אורך הבחינה הניחו כי:

sizeof (char) = 1
sizeof (int) = 4
sizeof (double) = 8
sizeof (void *) = 8
sizeof (size_t) = 8

שאלה 1 (10 נק')

נתון קוד בקובץ main.cpp הבא:

```
#include <iostream>

struct A {

      A()  { std::cout << " A ctor" << std::endl; }
      ~A() { std::cout << " A dtor" << std::endl; }

};

struct B : public A {

      B()  { std::cout << " B ctor" << std::endl; }
      ~B() { std::cout << " B dtor" << std::endl; }

};

struct C : public A {

      // data member
      B _b;

      C()  { std::cout << " C ctor" << std::endl; }
      ~C() { std::cout << " C dtor" << std::endl; }

};

int main () {
      C c;
      return 0;
}
```

הקוד מתקמפל. מה יהיה הפלט של הרצת התוכנית?

שאלה 2 (5 נק')

עבור כל הצהרה כתוב במחברת אם היא נכונה או לא **ונמק**.

א. מטודה וירטואלית יכולה להיות גם סטאטית

ב. קוד שכל הפונקציות שלו קומפלו כ inline תמיד ירוץ מהר יותר

שאלה 3 (25 נק')

נתון קטע הקוד הבא:

```
struct A {
      virtual void f()= 0;
      virtual void g() {}
      void h() {}
      virtual ~A() {}
};

struct B : public A {
      virtual void f(int i) {}
      virtual void g() {}
      virtual ~B() {}
};

struct C : public A {
      virtual void f() {}
      virtual ~C() {}
};

struct D : public B {
      virtual void f() {}
      virtual void h() {}
      virtual ~D() {}
};
```

כתוב במחברת את הטבלאות הוירטואליות של כל אחת מן המחלקות הנ"ל. שימו לב לסדר הירושות. הקפידו על הקידומת A:: או B:: או C:: **כמו כן**, במידה והפונקציה היא pure virtual הוסיפו סיומת 0= לאותה השורה בטבלה הוירטואלית. **בנוסף**, הקפידו על הסדר הנכון, כלומר, כך שהפונקציות הוירטואליות של הבן שהן גם וירטואליות אצל האבא יופיעו **באותו המקום בטבלה אצל הבן ואצל האבא, כולל ה destructors**, כך שאם מצביעים על בן ע"י מצביע לאבא הפונקציה נמצאת באותו המקום.

שאלה 4 (25 נק')

נתון קטע הקוד הבא:

```
1)    #include <iostream>
2)    #include <vector>
3)
4)    class GrayImage {
5)
6)    private:
7)     size_t _rows;
8)     size_t _cols;
9)     std::vector <unsigned char> _pixels;
10)    static unsigned int _numGrayImages;
11)
12)   public:
13)    GrayImage(size_t rows, size_t cols, unsigned char color= 0)
14)             : _rows(rows), _cols(cols), _pixels(rows*cols, color) {
15)      ++_numGrayImages;
16)    }
17)
18)    ~GrayImage() {
19)      --_numGrayImages;
20)    }
21)
22)    static unsigned int numGrayImages() {
23)      return _numGrayImages;
24)    }
25)
26)   };
27)
28)   unsigned int GrayImage::_numGrayImages= 0;
29)
30)   int main() {
31)
32)     std::cout << GrayImage::numGrayImages() << std::endl;
33)
34)     GrayImage gim(640,480);
35)
36)     std::cout << GrayImage::numGrayImages() << std::endl;
37)
38)     GrayImage* gimp;
39)
40)     std::cout << GrayImage::numGrayImages() << std::endl;
41)
42)     gimp= new GrayImage(320,240);
43)
44)     std::cout << GrayImage::numGrayImages() << std::endl;
45)
46)     delete gimp;
47)
48)     std::cout << GrayImage::numGrayImages() << std::endl;
49)
50)     return 0;
51)   }
```

א. (5 נק') רשום את פלט התוכנית.

ב. (16 נק') הטבלה הבאה מכילה כתובות. עליכם לציין לכל כתובת את המיקום שלה בזיכרון (בהתייחס לשלב הריצה כאשר התוכנית סיימה לבצע את מספר השורה שבסוגריים): מחסנית, גלובלי, ערימה דינמית, איזור הקוד, לא מוגדר. הניחו שכל ההקצאות הדינמיות מצליחות. כתבו במחברת לכל שורה בטבלה את המיקום (אין חובה להעתיק את הכתובת של השורה ובמקרה של טעות בהעתקת הכתובת נתייחס רק למספר השורה).

שורה	כתובת
1	`&gim (34)`
2	`&(gim._pixels) (34)`
3	`&(gim._rows) (34)`
4	`&(gim._pixels[0]) (34)`
5	`gimp (38)`
6	`gimp (42)`
7	`&(gimp->_rows) (42)`
8	`&(GrayImage::_numGrayImages) (42)`

ג. (4 נק') האם יש דליפת זיכרון בתוכנית?

שאלה 5 (25 נק'):

עליכם לכתוב פונקציה min_max_elements אשר מקבלת 2 סדרות.

אם הסדרות הן לא באותו האורך ו NDEBUG מוגדר, התנהגות הפונקציה לא מוגדרת (הכל יכול לקרות).

אם הסדרות הן לא באותו האורך ו NDEBUG לא מוגדר, הפונקציה תעצור את ריצת כל התוכנית ותתן הודעת שגיאה.

אם הסדרות הן באותו האורך, הפונקציה תשנה את הסדרה הראשונה כך שהיא תכיל במקום ה-i את המינימום מבין האיבר ה-i בסדרה הראשונה והאיבר ה-i בסדרה השנייה. את הסדרה השנייה היא תשנה כך שהיא תכיל במקום ה-i את המקסימום מבין האיבר ה-i בסדרה הראשונה והאיבר ה-i בסדרה השנייה. ראו דוגמא בתוכנית הבדיקה להלן.

הערות נוספות:

- חובה להשתמש ב assert ו std::swap במימוש הפונקציה.
- כתבו קוד גנרי ככל האפשר.
- במידה ו NDEBUG מוגדר, הפונקציה שלכם **לא צריכה** להשתמש בפרמטר הרביעי (בדוגמא פרמטר זה הוא vec.end()).
- אין צורך לכתוב includes.

```
#include "min_max_elements.hpp"
#include <forward_list>
#include <vector>
#include "gtest/gtest.h" // google test

TEST(minMaxElements, test1) {

 std::forward_list<int>         lst{ 1, 3, -6, 30};
 std::vector<int>               vec{-2, 1,  8, 20};
 std::forward_list<int> expectedLst{-2, 1, -6, 20};
 std::vector<int>       expectedVec{ 1, 3,  8, 30};

 min_max_elements(lst.begin(), lst.end(), vec.begin(), vec.end());

 EXPECT_EQ(expectedLst, lst);
 EXPECT_EQ(expectedVec, vec);
}
```

שאלה 6 (10 נק')

נתון קוד חלקי של המחלקה Complex. כתוב קוד של מחלקה גנרית של Complex. כלומר, כך ש Complex<double> יתנהג כמו Complex הנ"ל אבל יהיה ניתן גם לייצר אובייקטים מטיפוס Complex<float>. ניתן להניח כי מותר לכתוב re= 0.0 כאשר re הוא מהטיפוס הגנרי (כלומר, שיש המרה בין double לטיפוס הגנרי שמחזיק את המספרים במספר המורכב). אין צורך לממש אופרטורים או פונקציות אחרות שלא מופיעות בקוד להלן. עליכם לכתוב קוד מלא גם אם חלקו העתקה מהקוד להלן (אפשר לא להעתיק הערות).

```
class Complex {
private:
    double _re;
    double _im;
public:

    //-------------------------------------
    // Ctors & Destructor
    //-------------------------------------
    // Ctor works also as a conversion from double and
    // also as a default ctor.
    Complex (const double& re= 0.0,
             const double& im= 0.0)
        : _re(re), _im(im) {
    }
    //-------------------------------------

    //-------------------------------------
    // getters
    //-------------------------------------
    double re() const {
      return _re;
    }

    double im() const {
      return _im;
    }
    //-------------------------------------

    //-------------------------------------
    // operators
    //-------------------------------------

    //----------------------------------------
    // binary operators
    //----------------------------------------
    Complex& operator+=(const Complex& other) {
        _re+= other._re;
        _im+= other._im;
        return *this;
    }
};
```

בהצלחה!!!