שאלה 1

נתון קוד בקובץ main.cpp הבא:

```
A ctor
A ctor
B ctor
C ctor
C dtor
B dtor
A dtor
A dtor
```

שאלה 2

א. לא נכון. פונקציה סטטית היא למעשה פונקציה גלובלית ואין לה גישה לאובייקט. פונקציה וירטואלית צריכה גישה לאובייקט על מנת לקבל את האימפלמנטציה הנכונה (ה vptr שבתוך האובייקט).

ב. לא נכון. מכיוון שקובץ ההרצה יכול להיות גדול יותר וקח ייקח יותר זמן לעבור מאיזור קוד אחד לאיזור קוד אחר (פחות לוקליות).

שאלה 3

```
A virtual table:
A::f()= 0
A::g()
A::~A()

B virtual table:
A::f()= 0
B::g()
B::~B()
B::f(int)


C virtual table:
C::f()
A::g()
C::~C()

D virtual table:
D::f()
B::g()
D::~D()
B::f(int)
D::h()
```

שאלה 4 א

```
0
1
1
2
1
```

שאלה 4 ב

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

שאלה 4 ג

לא

שאלה 5

```
#pragma once
#include <algorithm>
#include <cassert>

template<typename InputIter1, typename InputIter2>
void min_max_elements(InputIter1 begin1, InputIter1 end1,
                      InputIter2 begin2, InputIter2 end2) {
 while (true) {
  assert( ((begin1!=end1) && (begin2!=end2))||
          ((begin1==end1) && (begin2==end2)) );
  if (begin1==end1) break;

  if ((*begin2)<(*begin1)) std::swap(*begin1, *begin2);

  ++begin1;
  ++begin2;
 }
}
```

שאלה 6

```
template<typename T>
class Complex {

private:

    T _re;
    T _im;

public:


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

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

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

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

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

בהצלחה!!!