Programming in C++ I
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH08-320142
November 7th, 2018

Practice Sheet

First Name:	....
Last Name:	....
Matriculation Number:	....

- Read all the following points before proceeding to the solution.
- Write immediately your name on this sheet.
- Write clearly. Take into consideration that C++ is a case sensitive language.
- Indent your code in a sensible way.
- Books, slides, notes or other documents are not allowed.
- If you need more space to solve the exercises you may use also the back of each page.
- Read carefully the questions and strictly adhere to the requirements.
- You have two hours to solve this test.
- Any attempt to cheat leads to an immediate fail.
- By signing this sheet you imply you read and understood all of the above.

Signature: ....

%	0.00 - 39.49	39.50 - 44.49	44.50 - 49.49	49.50 - 54.49
Grade	5.0	4.7	4.3	4.0

%	54.50 - 59.49	59.50 - 64.49	64.50 - 69.49	69.50 - 74.49
Grade	3.7	3.3	3.0	2.7

74.50 - 79.49	79.50 - 84.49	84.50 - 89.49	89.50 - 94.49	94.50 - 100.00
2.3	2.0	1.7	1.3	1.0

Problem P.1 *A Car* (6 points)

A car is characterized by a brand name, model name and a price. Write down the class `Car`. Its properties should not be directly accessible but rather be changeable by appropriate setter and getter methods, which should be implemented **inline**. Include constructors, destructors, setter and getter as inline methods.
Furthermore, it should be possible to create an instance of a car where all properties can be specified at object creation without calling further setting methods. Any user of this class should be able to set and read each property individually, but its properties should only be changeable by appropriate methods, which should be implemented **inline**. Please also include constructors and destructors.
Also, your class should provide a `print` method which prints all the information to the screen. Write also an **implementation** of the `print` method.

Problem P.2 *A taxi* (3 points)

Derive a class `Taxi` from the class `Car` you declared in the previous problem. A Taxi is a car which has a limit on the number of passengers it may carry. Write the **declaration and inline definition** of the class, including constructors, destructors, setter and getter methods. It should be possible to specify all properties of a Taxi at object creation. Also the class `Taxi` should provide a `print` method which prints all properties of a `Taxi` to the screen. Write also an **implementation** of the `print` method. The class declaration must be consistent with the one you provided while solving the previous problem.

Problem P.3 *Copy Constructor* (4 points)

Consider the declaration of the following classes: (please read carefully the given code):

```
class Book {
    private:
        const char* title;
        int pages;
    public:
        Book(const Book&);
        Book(const char* t, int p);
        Book();
        // ...
        // setter and getter methods omitted
};
```

a) Write an implementation of the copy constructor for `Book`

b) Normally a copy constructor will be automatically created by the compiler if none is provided. In short words, what does the compiler-created copy constructor do, and why should the programmer provide its own version for the class above?

Problem P.4 *Explanation* (1 point)

Please explain the keywords `private` and `protected` and point out their differences.

Problem P.5 *Overloading operators* (6 points)

Consider the following class (`complex.h`):

```
class Complex {

private:
    float real;  // real part
    float imag;  // imaginary part

public:
    Complex();
    Complex(float, float = 0);
    double magnitude();
    void print();

};
```

and the according definition of the print method in this fragment of complex.cpp:

```
void Complex::print() {
    if (imag)
        cout << noshowpos << real << showpos << imag << "i" << endl;
    else
        cout << noshowpos << real << endl;
}
```

(noshowpos does not display a plus sign if the number is positive, while showpos does)
Provide an overloaded << operator, which will replace the `print` function and creates the same output.
Also overload the < and > operators so you may compare complex numbers via their magnitude (so just use a `magnitude()` method for comparison (which you do —not— need to implement)).

Just add the declarations of the overloaded operators into `complex.h` above. Then just write down the definition below; it will be considered as part of the file `complex.cpp`.

Problem P.6 *A worker class* (4 points)

Write an appropriate class for the program below such that the class declaration, definition and this `main` function will compile and run together.

```
int main(int argc, char** argv) {
  worker a(234, "John McEnroe");
  worker b(324, "Jack Nicholson");

  cout << a << b;
  return 0;
}
```

Problem P.7 *Hierarchy of 3D-objects* (7 points)

Model and write down classes and their relationships (no diagram needed) corresponding to the following entities: `Object3D`, `Sphere`, `Cylinder`, `RectPrism` (i.e., rectangular prism), and `Cube`. An `Object3D` is characterized by a name and the rest are additionally characterized by their mathematical properties.
Make sure that the `Object3D` class is abstract with respect to a method called `volume()` which serves for computing and returning the volume of a 3D-object.
Implement constructor for setting properties, destructor and the `volume()` method for all the classes from above such that they will be consistent with the `main()` function from below.
The formulas for computing the volumes of the 3D-objects are:

$$V(Sphere) = \frac{4}{3} \cdot \pi \cdot radius^3$$

$$V(Cylinder) = \pi \cdot radius^2 \cdot height$$

$$V(RectPrism) = width \cdot length \cdot height$$

$$V(Cube) = side^3$$

For π you can use the `M_PI` macro from `cmath`.

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

// add your code here
```

```
int main(int argc, char *argv[]) {
    Object3D *arr[5];
    arr[0] = new Sphere("sphere", 1.3);
    arr[1] = new RectPrism("rectprism", 2.0, 3.0, 4.0);
    arr[2] = new Cylinder("cylinder", 2.0, 1.3);
    arr[3] = new Cube("cube", 2.0);
    arr[4] = new RectPrism("rectprism2", 1.0, 2.0, 3.5);
    int i;
    for(i=0; i<5; i++)
        cout << arr[i]->volume() << endl;
    for(i=0; i<5; i++)
        delete arr[i];
    return 0;
}
```

Totalpoints: 31