BERGISCHE
UNIVERSITÄT
WUPPERTAL

Prof. Dr. Hans-Jürgen Buhl
Praktische Informatik/Numerik

Fakultät für
Mathematik und Naturwissenschaften,
Mathematik und Informatik

E-MAIL buhl@math.uni-wuppertal.de
WWW www.math.uni-wuppertal.de/~buhl

DATUM 29. November 2017

Softwarequalität

WS 2017/2018 – Übungsblatt 6
Ausgabe: 29. November 2017
Abgabe bis 6. Dezember 2017 an: *mailto:Daniel.Schiller@uni-wuppertal.de*

Aufgabe 1. *LessThanComparable*

Auf den Seiten 12...13 von

http://www.diku.dk/forskning/performance-engineering/Generic-programming/Slides/concepts.pdf

wird das Requirement `LessThanComparable` mit den Axiomen `Transitivity`, `Antisymmetry`, `...` spezifiziert.

Erklären Sie in eigenen Worten, welche Eigenschaften hier semantisch gefordert werden. Vergleichen Sie mit http://www.sgi.com/tech/stl/LessThanComparable.html und Seite 431 von http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2011/n3242.pdf.

Aufgabe 2. *name_list*

Erläutern Sie in eigenen Worten die zweite und dritte Nachbedingung der Methode `name_list::put()` (Materialsammlung Seite 27).

Erweitern Sie die Vorbedingung auf `true` und ergänzen Sie zwei weitere Nachbedingungen der Form `ENSURE(not_in_list || ... ).`

Aufgabe 3. *bsearch*

Erläutern Sie natürlichsprachig in eigenen Worten den Codevertrag der Funktion `Iter bsearch(Iter p, Iter q, .)` von Seite 6 der Materialsammlung.

Aufgabe 4. *Kindklassen*

Können in C++ Kindklassen einzelne Attribute oder Methoden der Elternklasse nicht besitzen? Wenn ja: Wie kann das realisiert werden, und ist es sinnvoll? (Unterscheiden Sie C++03 und C++11)

Aufgabe 5. *structuredBinding*

Bringen Sie das Beispielprogramm

```
//============================================
// Name        : StructuredBindings.cpp
// Author      : HJB
// Copyright   : PD
// Description : structured bindings example
//============================================

#include <iostream>
#include <cstdlib>
#include <string>
#include <map>
//using namespace std;

int main(int argc, char *const argv[], char *const envp[]) {
        std::cout << "Test fuer structured Binding:\n" << std::endl;

        std::map<std::string, std::string> BUWStandorte = {
                        { "Campus Grifflenberg", "Gauss-Str. 20" },
                        { "Campus Haspel",       "Pauluskirchstr. 7" },
                        { "Campus Freudenberg",  "Rainer-Gruenter-Str." } };


        for (const auto& [key, value] : BUWStandorte){
                std::cout << key << ": " << value << std::endl;
        }
        // nach: https://www.heise.de/developer/artikel/C-17-Kleinvieh-macht-auch-Mist-3324790.

        // statt alt:
        std::cout << std::endl;
        for (const auto& it : BUWStandorte){
            std::cout << it.first << ": " << it.second << std::endl;
        }

        return EXIT_SUCCESS;
}
```

zum Ablauf. Welche Vorteile hat die erste for-Schleife verglichen mit der zweiten? Welcher gäben Sie den Vorzug?

Warum ist `return EXIT_SUCCESS;` der Anweisung `return 0;` vorzuziehen?

Was enthält `envp` in `main`?