SE 2 Klausur (1. Prüfungszeitraum)

Vorname: [redacted] Nachname: [redacted]

Matr.-Nr.: [redacted]

Dies ist mein dritter Prüfungsversuch ☐

- Bitte schalten Sie alle elektronischen Geräte aus.
- Es sind keine Hilfsmittel zugelassen.
- Die Klausur dauert 60 Minuten und es gibt insgesamt 60 Punkte.
- Bitte geben Sie die Klausur komplett und geheftet wieder ab.
- Bitte schreiben Sie mit schwarzem oder blauem Stift. Nicht mit Bleistift.
- Schreiben Sie bei Platzmangel auf der Rückseite weiter und markieren Sie dies.
- Bitte halten Sie ein Personaldokument und Ihren Studienausweis bereit.
- Bitte schreiben Sie deutlich! Nur Lesbares kann bewertet werden.

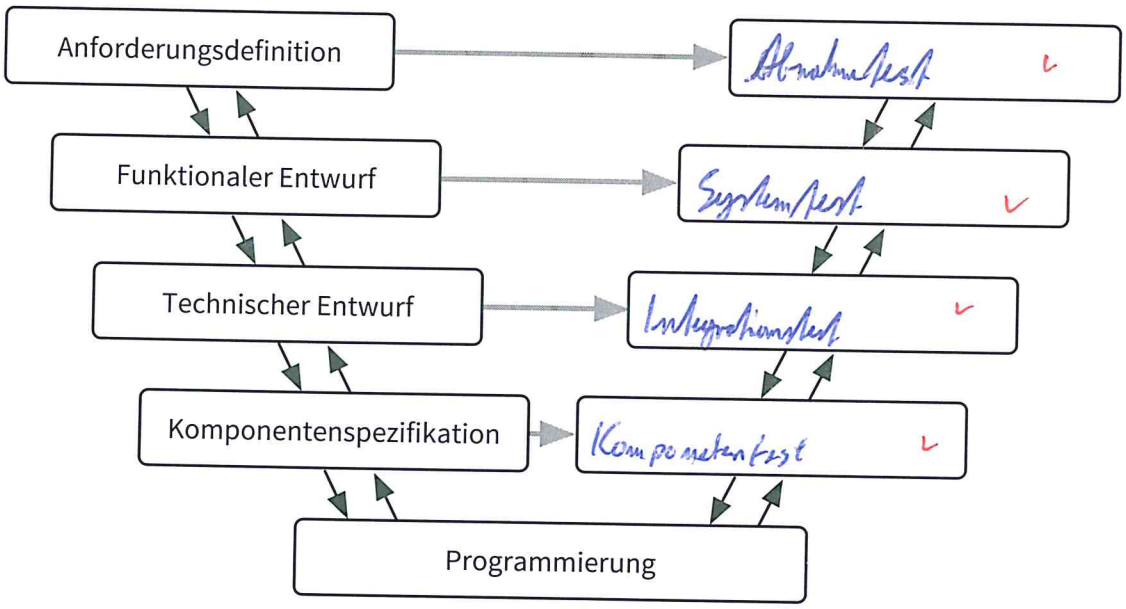
Folgende Tabelle ist für die Korrektur vorgesehen: Bitte nicht beschriften!

Aufgabe	1	2	3	4	5	6	7	8	9	10	Summe
Punkte	4	6	8	4	2	5	4	5	15	7	60
Erreicht	4	5	8	3,5	2	5	3	5	11,5	5,5	52,5

Note: 1,7

Aufgabe 1. 4 P.

Das folgende Bild zeigt das V-Modell der Softwareentwicklung. Tragen Sie in jedes leere Kästchen den Namen der jeweiligen Teststufe ein.



4

Aufgabe 2. 6 P.

Erklären Sie die Begriffe „Modularität", „Verteilung" und „Kopplung" mit ganzen Sätzen.

5

Modularität: (Modellebene)

Bedeutet, dass die Gesamtheit der Aufgaben an kleine Komponenten weiter aufgeteilt werden. Dadurch können z.B. Funktionen wiederverwendet werden und größere Strukturen werden verborgen. ✓ ?

Verteilung: (Modellebene)

Bedeutet, dass eine räumliche Trennung von Komponenten stattfindet. Dadurch kann die Performance gesteigert werden, da mehr Hardware verwendet werden kann ✓

Kopplung:

Ist eine Beschreibung der Abhängigkeit zwischen Komponenten z.B. Methodenaufrufe & Zugriff Gebiete. fn

Aufgabe 3. 8 P.

Nennen Sie für jede der vier genannten Kodierungsrichtlinien jeweils einen Vor- und einen Nachteil bei Verwendung.

	Vorteil	Nachteil
Schlumpfnamen benutzen	schnelles Erkennen, zu was die Variable/Klasse gehört ✓	Unnötig, [illegible] nur den Bezeichner, mehrere IDEs zeigen es an ✓
Selbsterklärende Namen verwenden	der Name lässt erkennen, was die Methode tut ✓	Alle Bezeichnungen werden länger ✓
Funktionen möglichst klein halten	übersichtlicher, keine God-Funktion ✓	man nicht voll durchschauen, da muss immer wieder [illegible] abgestiegen werden ✓
Yoda Conditions verwenden	Compiler weist einen auf die Zuweisung hin ✓	erfordert Umdenken, ✓

(8)

Aufgabe 4. 4 P.

Nennen Sie acht git Befehle:

- pull ✓
- push ✓
- fetch ✓
- merge ✓
- branch ✓
- commit ✓
- remove ✗ remote ?
- add ✓

(7½)

Aufgabe 5. 2 P.

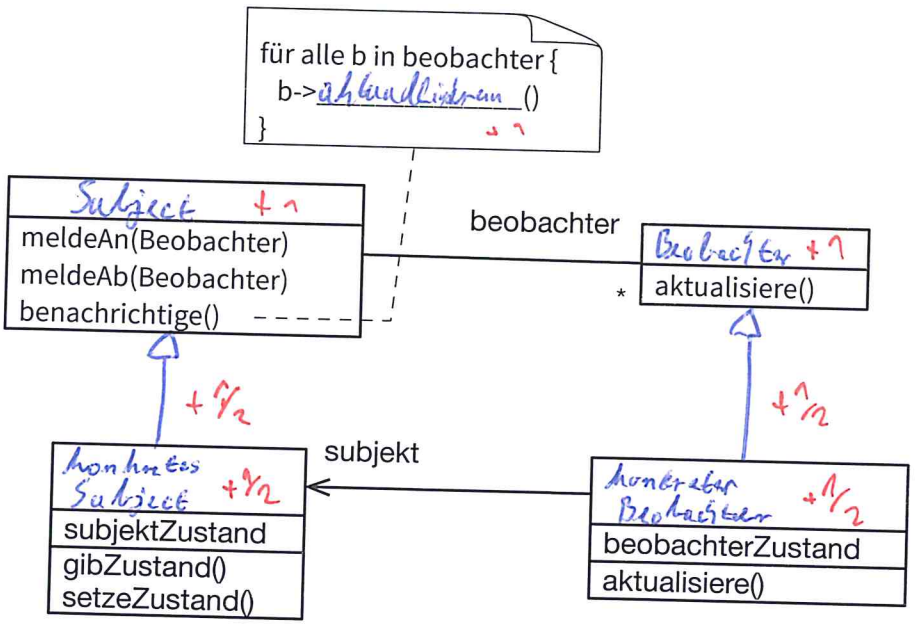
Vervollständigen Sie den folgenden Satz zu einer richtigen Aussage mit zwei von diesen drei Wörtern: Fehlerwirkungen, Fehlerzustände, Fehlhandlungen:

Statische Tests finden eher Fehlerzustände ✓ **als** Fehlerwirkung ✓.

(2)

Aufgabe 6. 5 P.

Hier ist das Entwurfsmuster „Beobachter“ abgebildet. Vervollständigen Sie das Diagramm hinsichtlich fehlender Klassennamen, Assoziationen und dem Inhalt der Notiz.



5

Aufgabe 7. 4 P.

Ein Tester kommt in eine Kneipe. Bestellt ein Bier. Bestellt 0 Bier. Bestellt 999999 Biere. Bestellt eine Eidechse. Bestellt -1 Bier. Bestellt ein sdkfjasbnföso...

Erklären Sie mit ganzen Sätzen was das mit dem Entwurf von konkreten Testfällen zu tun hat. Handelt es sich um einen guten Tester?

Der Tester hat hier für den Ablauf (Funktion) des Bestellens konkrete Parameter benutzt.

Ja, er probiert alle sinnvollen Eingaben aus. Wenn es sich jedoch nur um den Testfall des Bier bestellen handelt, dann wäre es unnötig eine Eidechse und ein sdkfj... zu bestellen.

3

Hinweis auf gültige / ungültige Werte fehlt

Was hat das mit konkreten TF zu tun?

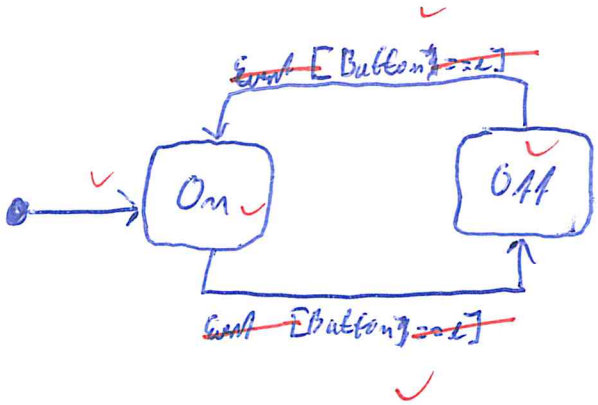
III

Aufgabe 8. 5 P.

Zeichnen Sie ein Zustandsdiagramm für folgenden C++ Quelltext:

```
enum State { On, Off };
enum Event { Button };

class StateMachine {
    State currentState_ = On;
public:
    State current_state() const { return currentState_; }
    void trigger(Event e) {
        switch (currentState_) {
        case On:
            if (e == Button) currentState_ = Off;
            break;
        case Off:
            if (e == Button) currentState_ = On;
            break;
        }
    }
};
```



5

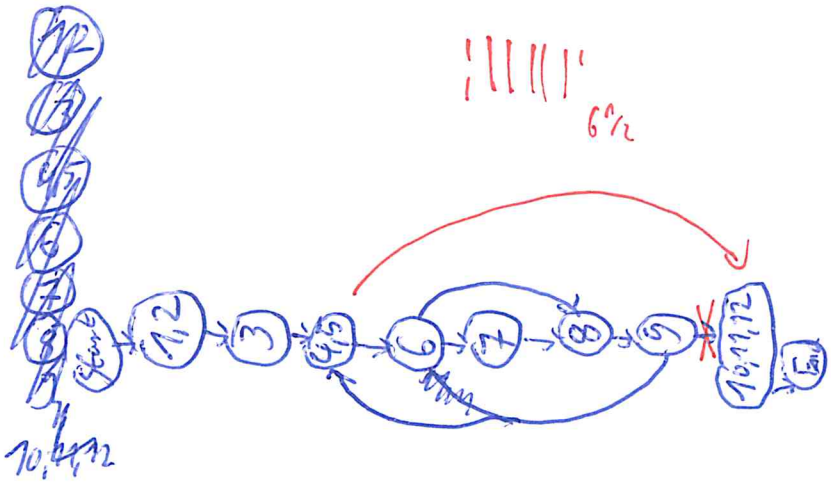
Aufgabe 9. 15 P.

Dieser C++ Quelltext berechnet die diskrete Exponentialfunktion:

```
1  int mod_pow(int base, int exponent, int modulus)
2  {
3      int result = 1;
4      while (exponent > 0)
5      {
6          if (exponent % 2 == 1)
7              result = (result * base) % modulus;
8          exponent = exponent >> 1;
9          base = (base * base) % modulus;
10     }
11     return result;
12 }
```

11 1/2

a) Zeichnen Sie den Kontrollflußgraphen. Markieren Sie Start- und Endzustände (8 P.)



Entwerfen Sie einen, oder gegebenenfalls mehrere abstrakte Testfälle für ...

b) eine komplette Anweisungsüberdeckung (2 P.)

{1,2; 3; 4,5; 6; 7; 8; 9; 10,11,12} ✓

c) eine komplette Entscheidungsüberdeckung (2 P.)

{1,2; 3; 4,5; 6; 7; 8; 9; 4,5; 6; 8; 9; 10,11,12} ✓

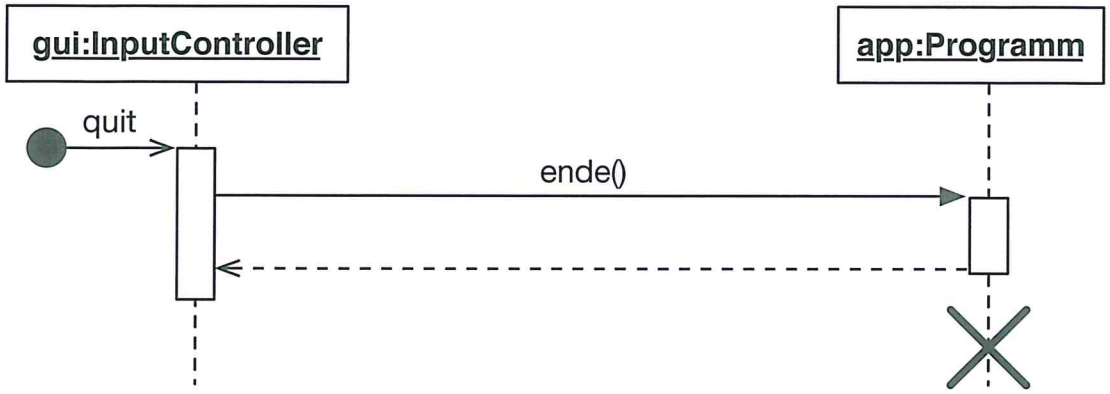
d) eine komplette Grenze-Inneres Überdeckung (3 P.)

{1,2; 3; 4,5; 6; 7; 8; 9; 4,5; 6; 9; 4,5; 6; 7; 8; 9; 10,11,12} +

+ 1

Aufgabe 10. 7 P.

Gegeben ist folgende Situation:



5 1/2

Entkoppeln Sie die Klassen **InputController** und **Programm** mittels des **Befehl** Entwurfsmusters. Vervollständigen Sie dazu in dem untenstehenden Sequenzdiagramm die Namen der Instanzen, sowie die Beschriftung der Aufrufe.