KLAUSUR im FACH ***OOSWE*** im Sommersemester 2023

Name, Vorname: …….

Studiengang/Semester: ……

Prüfer: Dipl.-Ing. (BA) Nico Klaas

Bearbeitungshinweise

1. Tragen Sie auf jeder Seite in der Kopfzeile Ihre Matrikelnummer ein.

2. Der Aufgabensatz (inkl. Deckblatt und Anhang), der aus 9 Seiten besteht (Seite 1 bis 9) ist auf Vollständigkeit zu überprüfen.

3. Der Aufgabensatz ist mit den Lösungsblättern abzugeben.

4. Lösungen auf selbst mitgebrachten Lösungsblättern werden nicht ausgewertet. Verwenden Sie die Ihnen ausgeteilten Lösungsblätter und tragen Sie auch dort Ihre Matrikelnummer ein.

5. Bei Rechenaufgaben muss der Lösungsweg ersichtlich und lesbar sein, sonst erfolgt keine Bewertung der Aufgabe oder des Aufgabenteils.

6. Die Bearbeitungszeit beträgt 60 Minuten.

7. Es wird hiermit darauf hingewiesen, dass vom Prüfungsamt nicht vorher geprüft wurde, ob Sie das Recht bzw. die Pflicht zur Teilnahme an dieser Klausur haben. Die Teilnahme erfolgt auf eigene Gefahr, gleichzeitig bekundet die Teilnahme die Zustimmung zu diesem Passus.

8. Hilfsmittel:
   - Coding Styleguide (ist selbst mitzubringen) – nur Markierungen mit einem Marker sind erlaubt (keine eigenen Notizen)
   - Taschenrechner sind **nicht** erlaubt – auch **keine** Smartphones und weitere digitale Geräte

9. **Die Nichteinhaltung des Coding Styleguides führt zu Punktabzug**

10. Bewertung:
    Gesamtpunktzahl: 69 Punkte (15% Überhang)
    Note 1,0: 60 Punkte Note 4,0: 30 Punkte

Aufgabe	**1**	**2**	**3**	**4**	**SUMME**
Punkte	**15**	**20**	**15**	**19**	**69**
Erreichte Punkte					

Aufgabe 1: (15 Punkte)

1.1 Kernaussage einer der ersten Folien war: C++ ist eine _____ Language. (1 P)

1.2 In C++ können auch C-Standardfunktionen verwendet werden. Allerdings sollten die korrespondierenden C++-Header verwendet werden (#include <cstring> statt #include <string.h>), da in den korrespondierenden C++-Headern die C-Standardfunktionen sich in automatisch im

_____ befinden. (1 P)

1.3 Implementieren Sie die Templatefunktion vSwap, welche die Inhalte der Referenzen tauscht. (2 P)

```
template <class T>
void vSwap(T& tra, T& trb)
{









}
```

1.4 Bei Unscoped Enums kann es in C++ zu _____ kommen. (1 P)

1.5 Gegeben sei die korrekte C-Funktion:

```
void foo42()
{
   Car* pCar = new Car();
   u32foo42Called++;
}
```

In welchem Speichersegmenten befinden sich:

Code der Funktion foo42: _____ (0,5 P)

Objektzeiger pCar: _____ (0,5 P)

Das allokierte Car-Objekt: _____ (0,5 P)

u32foo42Called: _____ (0,5 P)

1.6 Beim Most Vexing Parse wird eine vermeintlich statische Allokation eines Objektes vom Compiler

als _____ interpretiert. (1 P)

1.7 Beim Threading wurden zum Sperren des Mehrfachzugriffs bereits Mutexe wie in Ing.-Inf. behandelt. In C++ wurde die Verwendung von std::lock_guard vorgeschlagen.

```
std::lock_guard<std::mutex> LocalGuardObject(mutex_Counter);
```

Der grundsätzliche Vorteil gegenüber einem klassischen Mutex ist:

Es ist kein _____ notwendig, da dies im _____ des lock_guard-Objektes automatisch geschieht. (2 P)

1.8 Vervollständigen Sie die einfache Implementierung (in einer h-Datei) eines Smart Pointers. (3 P)

```
#pragma once
template<typename T>
class SimplePtr
{
public:
   SimplePtr(T* ptr) : _____________________
   {}

   ~SimplePtr()
   {
      ___________________________________
   }


   ___________ operator*()
   {
      return *ptr_;
   }


private:
   T* ptr_;
};
```

1.9 Es soll ein Smartpointer aus 1.8 mit dem Namen sptrA als Pointer auf ein Objekt der Klasse A verwendet werden. Das Objekt der Klasse A soll direkt bei der statischen Allokation von sptrA dynamisch allokiert werden. Vervollständigen Sie die Anweisung in foo.

```
void foo(void)
{
    SimplePtr_____________ sptrA (___________________________________);
}
```

(0,5 + 0,5 P)

1.10 Wann wird das in 1.9 dynamisch allokierte Objekt der Klasse A gelöscht? (Detaillierte Erklärung)

_____

_____

_____

(1 P)

Aufgabe 2: (19 Punkte)

```
#pragma once                                                    //Circle.h
#include <iostream>
typedef float f32_t;
class ____________________________;
std::ostream& operator<<(std::ostream& ostr, const Circle& Circle);

struct Point
{
   f32_t f32X_;
   f32_t f32Y_;
};

class Circle
{
public:
   Circle(Point PointTemp, f32_t f32RadiusTemp) noexcept;

   //Operator ++ (Präfix): Inkrement, Add 1 auf f32XRadius_, verkettbar, Original wird
verändert

   ____________________________________________________________
   //Operator +: Verschiebt PointCenter_ durch Addition von PointOffset
   //nicht verkettbar, Original wird verändert

   ____________________________________________________________
   //Operator +: Addiert zwei Circle, PointCenter_ ist Mittelwert, f32XRadius ist Summe
   //nicht verkettbar, Original wird nicht verändert

   ____________________________________________________________

private:
   Point PointCenter_;
   f32_t f32XRadius_;

   friend std::ostream& operator<<(std::ostream& ostr, const Circle& Circle);
};
```

2.1 Durch #pragma once wird die Headerdatei _____. (1 P)

2.2 Vervollständigen Sie Zeile 4. (1 P)

2.3 Per Default sind alle Attribute bei Strukturen in C++ automatisch: _____ (1 P)

2.4 Ergänzen Sie die Deklarationen der zu überladenden Operatoren. (3 x 1P)

2.5 Der + Operator gibt es nun zweimal. Der Unterschied ergibt sich durch die unterschiedlichen Übergabeparameter. Dies wurde im Script _____ von

_____ Operatoren genannt. (1 P)

2.6 Die Überladung des << Operators ist zwingend als friend-Funktion notwendig, da _____

_____ (1 P)

2.7 Die Anweisung `Circle* pc = new Circle{};` führt zu einem Compilerfehler, da

_____ (3 P)

2.8 Implementieren Sie drei Operatoren (++, +, +) in einer cpp-Datei. (10 P)
Verwenden Sie hierzu den Lösungsbogen

Aufgabe 3: (15 Punkte)

Gegeben sei das generische Observer Pattern aus der Vorlesung in der Push-Variante (siehe Anhang) Subject soll Sensor sein, Observer ist Anzeigepanel. Sensor enthält statt message_ jetzt f32SensorData_. Interfaces bleiben, bei Observer ändert sich nur der Klassenname.

3.1 Was ergibt sizeof(lSubject) auf einen 32-Bitsystem? _____ (1 P)

3.2 Ein set hätte im Gegensatz für die gegebene Problemstellung zu einer list den folgenden Vorteil:

_____ (1 P)

3.3 Eine list hat im Gegensatz zu set den folgenden Vorteil:

_____ (1 P)

3.4 In vDetach erfolgt kein delete. Entstehen hier keine Memory Leaks? (3 P)
Erklären Sie dies im Detail (ggf. anhand von RAII).

_____

_____

_____

3.5 Implementieren Sie in einer cpp-Datei die Funktion vDetach von Sensor. (3 P)

3.6 Implementieren Sie in einer cpp-Datei die Funktion vAttach von Sensor. (3 P)

3.7 Implementieren Sie in einer cpp-Datei die Funktion vNotify von Sensor. (3 P)

Aufgabe 4: (15 Punkte)

4.1 Gegeben sei:

```
class A
{
   uint32_t u32A;
protected:
   uint32_t u32B;
};
```

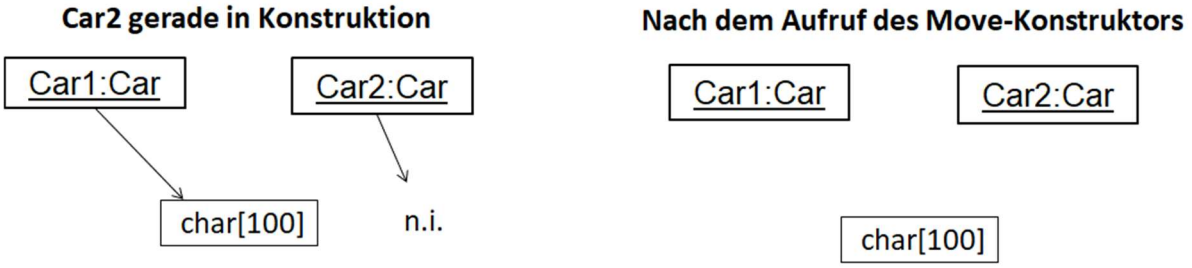
```
class B : [X] A
{
public:
   void foo(void)
   { /* some code */ }
};
```

Vervollständigen Sie die Tabelle mit ja/nein/vielleicht (kein Punktabzug bei Fehler) . (3 P)

	[X] = public	[X] = protected	[X] = private
Zugriff auf geerbtes u32A von foo aus			
Zugriff auf geerbtes u32B von foo aus			

4.2 Beim Kopierkonstruktur stellt sich nur dann die Frage: „Tiefe oder Flache Kopie?“ wenn die Klasse als Attribute _____ oder _____ enthält. (1 P)

4.3 Gegeben sei das folgende Objektdiagramm. Car2 wird durch `Car Car2 = std::move(Car1);` erstellt. Der eigene Move-Konstruktor soll die Daten an Car2 übergeben. Ergänzen Sie die rechte Abbildung aus dem Script. (2 P)

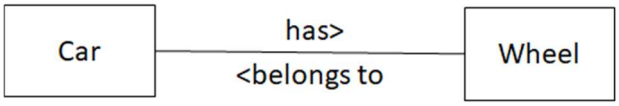


4.4 Erklären Sie exemplarisch anhand von C++ Code wie Exceptionhandling funktioniert. Das Werfen einer Exception muss nicht gezeigt werden. Zwei Schlüsselworte zum Exceptionhandling müssen vorhanden sein: (2 P)

4.5 Das Keyword noexcept ist für den _____ relevant. (1 P)

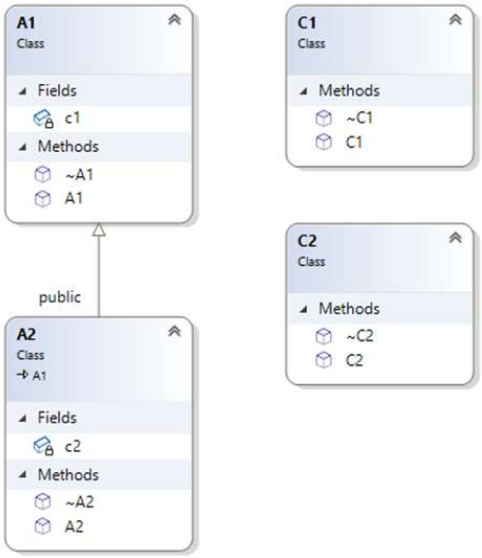
4.6 Mittels RTTI kann z.B. auf _____ zugegriffen werden. Dies kann bei dynamischem Polymorphismus nützlich sein. (1 P)

4.6 Fügen Sie dem Klassendiagramm die Multiplizität hinzu (ohne Reserverad, Wheel engl. = Rad) (2 P)
Die Beziehung ist als bidirektionale Assoziation dargestellt.



4.7 Welches **Prinzip** verletzt: (1 P)

```
A2* pa2 = new A1();
// some code
delete pa2;
```



4.8 Welcher Konstruktor wird durch new A1() zuerst aufgerufen. (1 P)

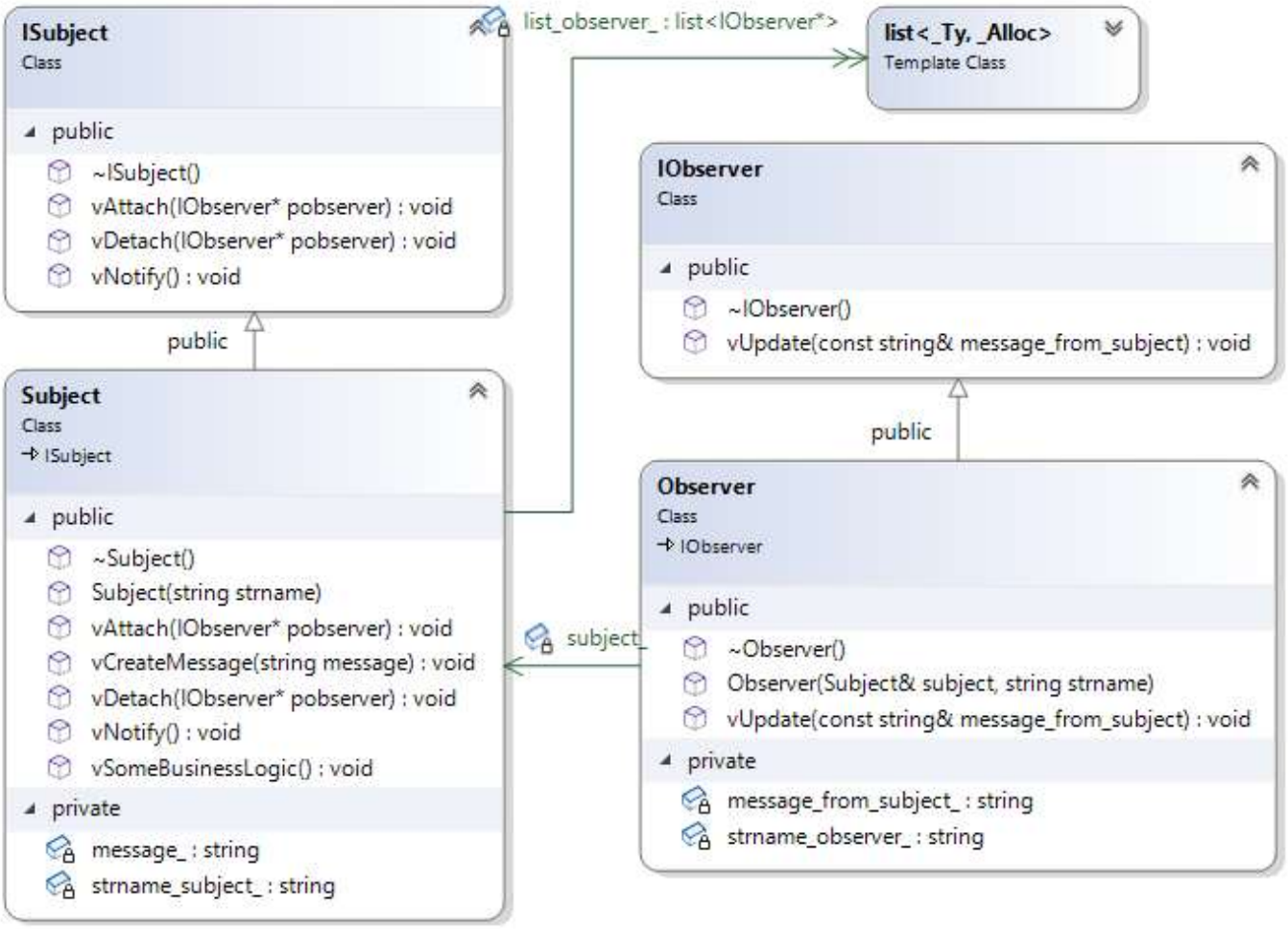
4.8 Wie muss Produktivcode in C++ aufgebaut sein, um diesen mit GoogleTest testen zu können? Zeigen Sie dies anhand eines Blockdiagramms auf! (2 P)

4.9 Ein Algorithmus hat eine Komplexität von O(n). Was bedeutet dies? (2 P)

4.10 Wie lautet die Komplexität von Bubblesort: O(_____) (1 P)

ANHANG:

Generisches Observer Design Pattern:



API list

Übergabeparameter der zu verwendenden Funktionen ergeben sich durch den Funktionsnamen.

Modifiers:

assign	Assign new content to container (public member function)
emplace_front	Construct and insert element at beginning (public member function)
push_front	Insert element at beginning (public member function)
pop_front	Delete first element (public member function)
emplace_back	Construct and insert element at the end (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
emplace	Construct and insert element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
resize	Change size (public member function)
clear	Clear content (public member function)

Operations:

splice	Transfer elements from list to list (public member function)
remove	Remove elements with specific value (public member function)
remove_if	Remove elements fulfilling condition (public member function template)
unique	Remove duplicate values (public member function)
merge	Merge sorted lists (public member function)
sort	Sort elements in container (public member function)
reverse	Reverse the order of elements (public member function)