Aufgabe 4: Gefäß-Klasse

(8 Punkte)

Schreiben Sie eine Klasse `gefaess`, die Folgendes leistet: Gefäße werden durch ihr Gesamtvolumen und das aktuell gefüllte Volumen charakterisiert. Diese beiden Variablen sollen von außen unzugänglich sein. Programmieren Sie folgende Methoden und Funktionen, so dass untenstehendes Programm funkioniert: (i) Setzen der Werte; (ii) Bestimmen des freien Volumens in einem Gefäß; (iii) Auffüllen eines Gefäßes von einem anderen Gefäß mit der größtmöglichen Menge; (iv) Vergleich der Füllmengen von zwei Gefäßen. Alle Volumenangaben haben dieselbe Einheit, z.B. Liter.
Die Klasse kommt in folgendem Programm zum Einsatz:

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

// Klasse gefaess und freie Funktion zu ergaenzen

int main()
{
    gefaess flasche( 1.0, 0.5) ; // Flasche mit Gesamtvolumen u. Fuellmenge
    gefaess glas( 0.3, 0.1) ;    // Glas mit Gesamtvolumen u. Fuellmenge

    cout << "Ins Glas passen noch " << frei(glas) << " Liter." << endl ;
    glas.fuelle_auf_von( flasche) ; // Fuelle das Glas aus der Flasche
                                    // auf, soweit dies moeglich ist
    if (flasche > glas)             // vergleiche die aktuellen Fuellmengen
       cout << "In der Flasche ist mehr als im Glas." << endl ;
}
```

Schreiben Sie die benötigte Klasse `gefaess` und alle benötigten Konstruktoren, Mitgliedsfunktionen und freien Funktionen, die in dem gegebenen Hauptprogramm aufgerufen werden.
Beachten Sie bitte beim Auffüllen, dass einerseits das Zielgefäß nicht überfüllt wird und andererseits höchstens die im Quellgefäß zur Verfügung stehende Volumenmenge verwendet werden kann. Bedenken Sie weiterhin, dass sich durch das Umfüllen auch die Menge im Quellgefäß ändert.
Der Rückgabewert des Vergleichs soll ein Logikwert sein.

Aufgabe 5: Ulam-Folge

(8 Punkte)

Die Ulam-Folge (u_i), $i \in \mathbb{N}$, benannt nach dem polnischen Mathematiker Stanisław Marcin Ulam, ist folgendermaßen definiert:

- $u_1 = 1$ und $u_2 = 2$
- für $n \geq 3$: u_n ist die kleinste natürliche Zahl, die sich *eindeutig* als Summe zweier *verschiedener* Zahlen aus $\{u_1, u_2, \ldots, u_{n-1}\}$ darstellen lässt.

Bsp: Die Folge beginnt mit den Zahlen: $1, 2, 3, 4, 6, 8, 11, \ldots$
$4 = 1 + 3$ ist die einzige Darstellung mit *verschiedenen* Summanden.
5 ist nicht in der Folge, da mit $5 = 1+4 = 2+3$ keine *eindeutige* Darstellung vorhanden ist.

Schreiben Sie ein vollständiges C++-Programm, das alle Folgenglieder u_n ausgibt, die kleiner als 1000 sind.

Kommentieren Sie die einzelnen Abschnitte in Ihrem Programm, so dass sofort verständlich ist, was gemacht wird. Ansonsten kann der entsprechende Programmteil nicht bewertet werden.

Aufgabe 1: Zwei mal Minimum

(4 Punkte)

(a) Schreiben Sie eine Funktion `min`, die zwei `int`-Argumente hat und das Minimum der Zahlen ermittelt und zurückgibt.
(b) Überladen Sie nun die Funktion `min`, indem Sie eine Variante mit drei `int`-Argumenten programmieren. Allerdings darf in dieser Funktion keine explizite Fallunterscheiung auftauchen, sondern die Berechnung soll mittels Aufrufen der Funktion aus Teil (a) erfolgen.

Aufgabe 2: Hamming-Distanz

(5 Punkte)

In C++ gibt es eine Klasse `string` zum Arbeiten mit Zeichenketten in der Standardbibliothek. Sie stellt unter anderem folgende Methoden zur Verfügung: (i) `int size()` ergibt die Länge einer Zeichenkette und (ii) `char & at(int p)` liefert das Zeichen, das an der Position p der Zeichenkette steht. Der Postionsbereich p, in dem die Zeichen abgespeichert sind, geht dabei von 0 bis zur (*Länge der Zeichenkette* - 1).
Schreiben Sie damit eine Funktion `hamming`, die die Hamming-Distanz von zwei Zeichenketten ermittelt. Die Hamming-Distanz ist definiert als Anzahl der Positionen, an denen zwei gleich lange Zeichenketten voneinander abweichen. Die Funktion habe zwei `string`-Argumente und einen `int`-Rückgabewert.

Sie dürfen davon ausgehen, dass die Funktion immer zwei gleich lange Zeichenketten erhält.

Bsp: `tuba` und `tUbe` haben die Distanz zwei, weil das zweite und vierte Zeichen verschieden sind.

Aufgabe 3: Was wird ausgegeben?

(5 Punkte)

Setzen Sie an der markierten Stelle in folgendem Programm die letzten drei Ziffern Ihrer Matrikelnummer ein. Schreiben Sie dann *vollständig* auf, was von dem Programm ausgegeben wird.

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

void f( int * z)
{
    (*z) += 10 ;
    cout << "Rom: " << *(z+1) << endl ;
}

void f( int & z)
{
    z += 20 ;
    cout << "Paris: " << z << endl;
}

int main()
{
    int a[3] = {_, _, _} ; // <- setzen Sie hier bitte die *drei letzten*
                           //    Ziffern Ihrer Matrikelnummer ein.

    cout << "Oslo: " << a[0] << " " << a[1] << " " << a[2] << endl ;
    f(a[2]) ;
    f(a) ;
    f(&(a[1])) ;
    f(*(a+1)) ;
}
```