Gebt bitte die Bearbeitung als .pdf oder .doc Datei an Tobias Hammer (`hammer@informatik.uni-bremen.de`) ab. Bitte stellt die Abgaben zu allen Aufgaben in einer Datei zusammen, die wir drucken und korrigieren können. Der Dateiname soll die Nummer des Übungszettels und der Gruppe beinhalten, z.B. `ifguebung407.pdf` für Übung Nr.4 Gruppe 7. Gebt Namen und Email aller Gruppenmitglieder an. Gebt ausserdem die Zeit (Maximum aller Gruppenmitglieder) an, die Ihr für den Zettel benötigt habt. Diese Angabe ist nur als Rückmeldung für uns und geht nicht in die Bewertung ein.

Aufgabe 13 Schnipsel (5 Punkte)

Aufgabe 13.1 Funktionen (1 Punkt)

Ihr habt folgenden Codeschnipsel gegeben. Schreibt auf, was das Programm ausgibt **und** schreibt zu jeder Ausgabe kurz, warum sie genau mit den Zahlen an dieser Stelle geschieht.

```
void fkt1( int a, int b )
{
  a = 3;
  b = a + b;
  println( "fkt1: " + a + "  " + b );
}

int fkt2( int a, int b )
{
  fkt1( a, b );
  a = 10;
  println( "fkt3: " + a + "  " + b );
  return b + 1;
}

int fkt3( int a, int b )
{
  println( "fkt4: " + a + "  " + b );
  return fkt2(a, a) + b;
}

int a = 1;
int b = 2;
fkt1(a,b);
println( a + "  " + b );
a = fkt2(a,b);
println( a + "  " + b );
b = fkt3(a,b);
println( a + "  " + b );
```

Aufgabe 13.2 Mathe → Processing (1 Punkt)

Dieses mal sollt ihr ein wenig Mathe in Form von zwei einfachen Formeln in ein Processing-Programm gießen. Es geht darum zu berechnen, wo ein Ball den Boden berührt, wenn er physikalisch korrekt fällt. Dafür sind eine X- und Y-Position gegeben, an denen er abgeworfen wird und eine Geschwindigkeit in X-Richtung. Ziel ist es nun, herauszufinden, wie lange der Ball braucht, bis er den Boden berührt (er wird durch die Schwerkraft Richtung Boden beschleunigt) und wie weit er in dieser Zeit in X-Richtung fliegt.

Die zugrunde liegende Formel benötigt ihr nicht, ist der Vollständigkeit halber aber angegeben:

$$y(t) = y_{start} + v_{ystart} \cdot t + \frac{1}{2} \cdot g \cdot t^2$$
$$x(t) = x_{start} + v_{xstart} \cdot t$$

dabei ist s die Position, v die Geschwindigkeit, g die Erdbeschleunigung und t die Zeit.

Ein kleines Rahmenprogramm ist euch vorgegeben:

```
void setup() {
  size(400,400);
}

float impactPositionX( float x, float y, float vx ) {
  // TODO: program
  return 0; // delete this line
}

void draw() {
  float xStart = mouseX, yStart = mouseY, vx = 25, radius = 15;  // Parameters
  background(255, 255, 255);
  stroke(255, 0, 0);
  float y = yStart, x = xStart, timeStep = 0.1, vy = 0;
  while ( y < height ) {
    ellipse( x, y, 1, 1 );
    y  += vy*timeStep+9.81*sq(timeStep);
    x  += vx*timeStep;
    vy += +9.81*timeStep;
  }
  fill(0, 0, 0); stroke(0, 0, 0);
  float xEnd = impactPositionX( xStart, yStart, vx );
  ellipse( xEnd, height, 2*radius, 2*radius );
}
```

Eure Aufgabe ist es, die Funktion `impactPositionX` zu füllen. Dafür müsst ihr folgende Formel in Code umsetzen. Sie berechnet im hinteren Teil ($\sqrt{\ldots}$) die Zeit bis zum Boden und setzt diese dann in die oben angegebene Formel ein. g ist die Erdbeschleunigung (Schwerkraft) und ist $9,81$.

$$x_{impact} = x + v_x \cdot \sqrt{\frac{2 \cdot (height - y)}{g}}$$

Zur Überprüfung ist die Ballflugbahn in Rot eingezeichnet. Euer Endpunkt sollte an der Position sein, an dem die gepunktete Linie in den Boden einschlägt (impact).

Aufgabe 13.3 Bilder (1 Punkt)

Schreibt ein kleines Processing-Programm, dass ein Bild lädt und darstellt. Das Bild soll dem Mauszeiger folgen und zwar so, dass sich die Bildmitte immer an der Mausposition befindet. Normalerweise soll es 50x50 Pixel groß sein, nur wenn eine Maustaste gedrückt wird, soll das Bild 100x100 Pixel groß sein. Auch dann sollte das Bild noch um den Mauszeiger zentriert dargestellt werden

Aufgabe 13.4 Ein Klasse für sich (2 Punkte)

In dieser Aufgabe sollt ihr eine kleine Klasse schreiben, deren Objekte je ein Monster aus einem typischen Computerspiel repräsentieren. Überlegt euch, welche Methoden solch eine Klasse benötigt und welche Parameter diese haben könnten. Überlegt ausserdem, welche Variablen solch ein Monster-Objekt benötigen könnte. Schreibt die Klassenbeschreibung in processing auf. Ihr müsst weder ein Programm schreiben, das euer Monster verwendet, noch müsst ihr die Methoden programmieren. Schreibt aber in jeder Methode einen kurzen Kommentar, was diese tut, wie sie ihre Parameter verwendet und was sie für Variablen verändern kann. Es reichen ca. 4 Methoden und 5-6 Variablen aus.

Aufgabe 14 Mein erstes objektorientiertes Computerspiel (8 Punkte + 2 Kreativpunkte)

In dieser Aufgabe soll Euer Computerspiel von Aufgabe 7 (Zettel 2) mit Hilfe von Funktionen, Objekten und Methoden neu strukturiert und erweitert werden. Unterteilt dazu Euer Programm in ein kleines Hauptprogramm (`setup` und `draw`) und mehrere Klassen. Als grobe Richtlinie kann jede sich im Spielablauf bewegende Figur (Spieler, Ball, etc.) ein Objekt einer Klasse sein. Außerdem kann häufig das Spielfeld eine Objekt einer weiteren Klasse sein, das hängt vom Spiel ab.

In einem sauberen Design greifen normalerweise nur die Funktionen einer Klasse auf die Variablen dieser Klasse zu. Zugriffe von außen sind der Ausnahmefall. Entwickelt Euer Programm gemäß dieser Richtlinie. Ein häufiger Ausnahmefall ist eine Funktion die überprüft, ob zwei Spielfiguren kollidieren. Sind die Spielfiguren aus verschiedenen Klassen ist unklar Methode welcher Klasse diese Funktion sein sollte. Dann ist die einzige Lösung eine Funktion, der beide Objekte als Parameter übergeben werden. Diese muss dann zwangsweise auf deren Variablen zugreift.

Überarbeitet auch Euer Spielkonzept. Beseitigt Probleme die sich herausgestellt haben und erweitert das Spielprinzip, so dass von einer Klasse, also einem Typ Spielfigur mindestens zwei Exemplare im Spiel sind. Realisiert dies ohne doppelte Programmabschnitte, sondern durch zwei Objekte einer Klasse.

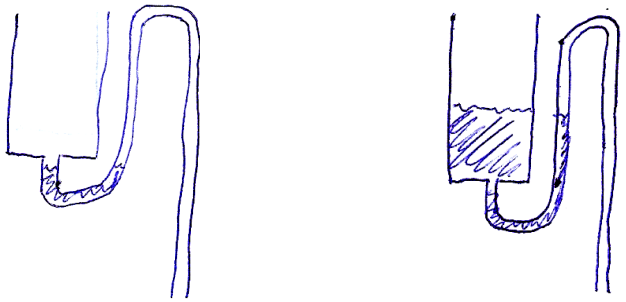

Abbildung 1: Saugheber als zentrales Element des Wasserschwall-Computers. *Links*: Der leere Saugheber stellt eine `0` dar. *Rechts*: Wasser im Saugheber stellt eine `1` dar. Fließt ein weiterer Schwall Wasser zu, übersteigt der Pegel das obere Ende des U-Rohrs. Durch den Saugeffekt wird der ganze Behälter leer gesaugt (`1`→`0`).

Aufgabe 15 Der Wasserschwall-Computer (2 Bonuspunkte)

In der Geschichte der Informatik wurden die selben Rechenoperation, wie z.B. Addition, auf verschiedenen Technologien realisiert: Zuerst mechanische Zahnräder gefolgt von elektro-mechanischen Relais, Elektronenröhren, Transistoren und Mikrochips. Im Prinzip denkbar wäre auch eine Konstruktion aus wassergefüllten Behältern und Röhren [1]. Ein zentrales Element ist dabei der Saugheber (Abb. 1). Der Behälter speichert Wasser, wobei kein Wasser `0` und Wasser vorhanden `1` bedeutet. Durch ein umgedrehtes U-Rohr entleert sich der Saugheber vollständig, sobald sein Wasserstand die Oberkante des U erreicht.

Konstruiert aus diesem Baustein einen Binärzähler mit 4 Binärstellen (0000, 0001, 0010, 0011, 0100, ..., 1110, 1111) Gebt eine Skizze und eine Beschreibung ab.

[1] Diese Aufgabe wurde inspiriert durch die Wasseruhr im Berliner Europazentrum (`http://www.time-flow-clock.de/`).