Aufgabe 1: Pellgleichung (5 Punkte)

Die sogenannte Pellgleichung

$$x^2 - d \cdot y^2 = 1 \quad \text{mit} \quad d \in \mathbb{N} \text{ und } x, y \in \mathbb{Z}$$

ist ein Problem aus der Zahlentheorie. Für gegebenes positives ganzzahliges d werden ganzzahlige Lösungspaare x, y gesucht.
Schreiben Sie ein vollständiges C++-Programm, das Folgendes leistet: Der Benutzer kann eine positive ganze Zahl d eingeben, es braucht dabei nicht überprüft zu werden, ob $d > 0$ ist. Sodann sollen im Wertebereich $x, y = 0, \ldots, 100$ alle Lösungen der obigen Gleichung durch simples Ausprobieren von x und y ermittelt und ausgegeben werden. Weiterhin soll die Anzahl der gefundenen Lösungen bestimmt und ausgegeben werden.

Aufgabe 2: Nullfeld? (6 Punkte)

(a) Schreiben Sie eine C++-Funktion `is_zero`, die untersucht, ob in einem Feld von n `double` Variablen alle Werte exakt null sind. Die Funktion soll die Länge n des Felds und das Feld selbst als Argument erhalten. Gültige Feldindizes sind im Bereich $0, \ldots, n-1$.
Der Rückgabewert soll ein Logikwert sein, der „wahr" ist, wenn alle Elemente null sind. Für Felder der Länge $n = 0$ soll als Ergebnis auch „ist ein Nullfeld" herauskommen.

(b) Schreiben Sie eine rein rekursive Variante `is_zero_rec`, die die gleiche Aufgabe ohne explizite Schleife erledigt. Die Funktion soll die gleichen Argumente und den gleichen Rückgabewert wie in Teil (a) haben.
Tipp: Überlegen Sie sich hierzu, wie Sie die Aufgabenstellung durch separate Behandlung des letzten Feldelements und des Restfeldes betrachten können.

Aufgabe 3: Was wird ausgegeben? (4 Punkte)

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

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

int f(int* p) {
    (*p)+=3 ;
    return 7/2 ;
}

int f(int& z) {
    z += 5 ;
    return z*2 ;
}

int main() {
    int i1= 6 ; // <- setzen Sie hier bitte die letzte Ziffer
                //    Ihrer Matrikelnummer ein.
    int i2= i1 ;

    int j=f(i1) ; i1=11   j=22
    cout << "Alpha: " << i1 << " " << j << endl ;

    j=f(&i2) ;  i2=9
                j=3
    cout << "Beta: " << i2 << " " << j << endl ;
}
```

Aufgabe 4: 2d-Vektoren-Klasse (8 Punkte)

Eine Klasse soll zu Rechnungen mit 2-dimensionalen Vektoren dienen. Die Komponenten der Vektoren sollen von außen unzugängliche `double`-Variablen sein.
Folgende Methoden bzw. freie Funktionen sollen vorhanden sein:
(i) Konstruktor ohne Argument.
(ii) Konstruktor mit zwei Argumenten, der die Vektorkomponenten initialisiert.
(iii) Addition zweier Vektoren.
(iv) Normierung (mit Euklidnorm) eines Vektors auf die Länge 1, ändert den Vektor selbst.
(v) Skalarprodukt zweier Vektoren.

Die Klasse kommt in folgendem Programm zum Einsatz:

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

// Klasse vec2d und freie Funktion zu ergaenzen

int main()
{
    vec2d v1 (1.0, 2.0) ;
    vec2d v2 (3.0, -4.0) ;
    vec2d sum = v1+v2 ;    // Summe zweier Vektoren berechnen
    sum.norm() ;           // Vektor auf Laenge 1 normieren
    cout << "Das Skalarprodukt von v1 und v2 ist: " << skp(v1,v2) << endl ;
}
```

Schreiben Sie die benötigte Klasse `vec2d` und alle benötigten Konstruktoren, Mitgliedsfunktionen und freien Funktionen, die in dem gegebenen Hauptprogramm aufgerufen werden. Die Definition der Mitgliedsfunktionen soll jeweils außerhalb der eigentlichen Klassendefinition erfolgen. Die Wurzel einer Zahl kann mittels `sqrt(.)` berechnet werden.

Aufgabe 5: Zerlegung in zwei Primzahlen (7 Punkte)

Die Goldbachsche Vermutung lautet: *Jede gerade Zahl größer als 2 kann als Summe zweier Primzahlen geschrieben werden.* Diese Vermutung ist bis heute unbewiesen.
Schreiben Sie ein Programm, welches Folgendes leistet: Zu einer eingegebenen Zahl sollen alle möglichen Zerlegungen der Zahl in zwei Primzahlsummanden ermittelt und ausgegeben werden. Sie brauchen nicht zu überprüfen, ob die eingegebene Zahl größer als 2 und gerade ist.

Bsp: Eingabe: 14 $\longrightarrow$ Ausgabe: 3+11 7+7.

Kommentieren Sie die einzelnen Abschnitte in Ihrem Programm, so dass sofort verständlich ist, was gemacht wird!

Hinweise: (i) 1 ist keine Primzahl. (ii) Der Operator `%` ergibt den Rest einer Ganzzahl-Division.
