WS07/08

Aufgabe 1: Teilersumme (4 Punkte)

Die Teilersumme einer Zahl ist definiert als Summe aller Ihrer Teiler. Beispiel: Die Teilersumme von 10 ist 18, wegen der Teiler 1, 2, 5 und 10.
Schreiben Sie ein vollständiges C++-Programm, das von der Tastatur eine `long`-Zahl einliest, die Teilersumme berechnet und ausgibt. Sie dürfen voraussetzen, dass die eingegebene Zahl größer null ist. Hinweis: Der Operator für den Divisionsrest ist `%`.

Aufgabe 2: Felder vergleichen (4 Punkte)

Schreiben Sie eine Funktion, die überprüft, ob zwei Felder mit ganzzahligen Einträgen im Indexbereich $0, \ldots, n-1$ identisch sind.
Die Funktion soll als Argument die beiden Felder mit dem Grunddatentyp `int` und die Größe der Felder n erhalten. Das Ergebnis ist als Logikwert von der Funktion zurückzugeben.

Aufgabe 3: Klasse für Polarkoordinaten (6 Punkte)

Schreiben Sie eine Klasse `polar2d`, die zur Beschreibung von Punkten in zwei Dimensionen dient. Ein Punkt soll dabei durch Polarkoordinaten (r, α) dargestellt werden. Der Winkel α wird dabei in üblicher Weise im Bogenmaß gegen den Uhrzeigersinn ab der positiven x-Achse gemessen.
Die Klasse soll in folgendem Programmteil zum Einsatz kommen:

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

// Klasse polar2d zu ergaenzen

int main()
{
    const double pi=3.141592654 ;

    polar2d p1 (1.0, pi/2) ; // vereinbare einen Punkt durch Radius und Winkel
    p1 *= 2.0 ;              // aendere Radius von p1 um angegebenen Faktor;
    p1.rotate(-pi/6) ;       // rotiere p1 um das Winkelargument ;
    cout << "Kartesische X-Koordinate: " << get_x (p1) << endl ;
}
```

Schreiben Sie die benötigte Klasse `polar2d` und alle benötigten Konstruktoren, Mitgliedsfunktionen und freien Funktionen, die in dem gegebenen Hauptprogramm aufgerufen werden. Der Radius und die Winkelvariable sollen dabei von außen unzugängliche `double`-Variablen sein. Sie brauchen bei beiden Variablen der Einfachheit halber keine Bereichsüberprüfungen vorzunehmen. Die Definition der Methoden soll außerhalb der eigentlichen Klassendefinition erfolgen. Die beide Methoden `*=` und `rotate` sollen jeweils den Punkt selbst ändern, beide Methoden haben jeweils ein `double` Argument. Die freie Funktion `get_x` soll die x-Koordinate des Punktes berechnen.

Aufgabe 4: Trapezregel (3 Punkte)

Mit der Trapezregel kann ein Integral einer Funktion numerisch näherungsweise berechnet werden:

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\Big(f(a)+f(b)\Big) + \text{Restglied}.$$

Schreiben Sie eine C++-Funktion, die als Argument einen Funktionszeiger und zwei Integrationsgrenzen a und b erhält und als Funktionswert die oben angegebene Trapezsumme berechnet. Die zu übergebende Funktion hat dabei folgende Signatur: `double f(double)`.

Aufgabe 5: Was wird ausgegeben? (5 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 ;

void f(int* q) {
    int i7 = *q ;
    (*q) += 3 ;
    i7   += 2 ;
    cout << "Fkt1: " << i7 << endl ;
}

void f(int& q) {
    q += 1 ;
    cout << "Fkt2: " << q << endl ;
}

int f(int r, int s) {
    if (r<=s) return 0 ;
    return (r-s) + f(r-1,s+1) ;
}

int main() {
    int i1= 5 ; // <- setzen Sie hier bitte die letzte Ziffer
                //    Ihrer Matrikelnummer ein.
    int i2 = i1 ;
    f(i1) ;
    cout << "Main1: " << i1 << endl ;
    f(&i2) ;
    cout << "Main2: " << i2 << endl ;
    cout << "Main3: " << f(7,3) << endl ;
}
```

Aufgabe 6: Mirpzahlen (8 Punkte)

Mirpzahlen sind Zahlen, die vorwärts und rückwärts gelesen Primzahlen sind. Bsp: 13 ist Mirpzahl, da sowohl 13 als auch 31 prim sind.
Schreiben Sie ein vollständiges C++-Programm, das für eine eingegebene `long`-Zahl testet, ob sie Mirpzahl ist. Sie brauchen nicht zu überprüfen, ob die eingegebene Zahl positiv ist.

Hinweis: Durch Programmierung geeigneter Funktionen lässt sich die Aufgabe in Teilprobleme zerlegen.