Ziele sind das Arbeiten mit Funktionen, Parameter, Rückgabewerte, Aufzählungstypen

Vorbemerkung:
Diese Aufgabe baut auf der 3. Übung auf und erweitert diese um weitere Funktionen. Sie benötigen also eine funktionierende Lösung von Übung 3. Diese muss nicht perfekt sein, aber in den Grundzügen funktionieren.

Schritt 0 (analog zu Übung 3):
Starten Sie die Entwicklungsumgebung und öffnen Sie das Projekt mit der letzten Übung. Fügen Sie dann der Projektmappe ein neues Projekt **hinzu** (keine neue Projektmappe anlegen!). Legen Sie dann im neuen Projekt eine leere C++-Datei (etwa aufg4main.cpp) an. Legen Sie anschließend (rechte Maustaste auf den Namen des neuen Projektes) das neue Projekt als **Startprojekt** fest. Zum Abschluss der Vorbereitungen **kopieren** Sie den Inhalt (Quelltext) der 3. Übung in die leere Quelltextdatei der neuen Übung. Sie haben damit eine **exakte Kopie** von Übung 3 erstellt, mit der Sie jetzt weiter arbeiten. **Überprüfen Sie die Funktion des neuen Projektes!**

Schritt 1:
Fügen Sie dem neuen Projekt eine Headerdatei (z.B. funktionen4.h) hinzu. Kopieren Sie in diese Datei den Quelltext vom Ende des Aufgabenblattes bzw. aus der Datei aus Moodle bzw. der Webseite. Ersetzen Sie dann die ersten Zeilen der cpp-Datei durch einen #include "funktionen4.h", testen Sie erneut die Funktionalität!.

Danach fügen Sie eine weitere Quelltextdatei (z.B. funktionen4.cpp) zum Projekt hinzu. Auch in diese Datei muss die Headerdatei eingebunden werden. Damit sind die Vorarbeiten abgeschlossen.

Aufgabe 4 Teil 1:
Bauen Sie jetzt systematisch das Programm aus Übung3 so um, dass die einzelnen Menüpunkte durch eine (oder auch mehrere) Funktionen ersetzt werden. Beginnen Sie mit einer Funktion für das Osterdatum, dann für das Schaltjahr und dann für die p-q-Formel. Beachten Sie dabei die Vorgaben und Erläuterungen in der vorgegebenen Headerdatei, **diese sind einzuhalten**. Dabei darf die Headerdatei nur geändert werden, wenn Sie das **vorher** mit dem Dozenten abgesprochen haben. Achten Sie insbesondere darauf, an welcher Stelle (Hauptprogramm oder Funktion) Werte ein- oder ausgegeben werden. Prüfen Sie nach jeder neuen Funktion, ob Ihr Programm noch so funktioniert und protokollieren Sie (schriftlich auf Papier) ihre diesbezügliche Tests!

Wenn diese drei Funktionen korrekt umgesetzt sind geht es mit der Warteschlangensimulation weiter. Hier erfolgt die Umsetzung durch **drei** Funktionen, die jeweils einen Teil der Aufgabe übernehmen (anfügen, entfernen und ausgeben). Auch hier gilt die ausführliche Beschreibung in der Headerdatei. Damit das klappt, müssen Sie eventuell die Ablauflogik überprüfen und entsprechend anpassen.

Zum Abschluss dieses Teils wird der noch fehlende Menüpunkt „Dreiersummen“ in eine Funktion umgesetzt. Diese Funktion kommt völlig ohne Parameter und Rückgabewerte aus, darf aber auch keine Variablen aus dem Hauptprogramm verwenden.

Aufgabe 4 Teil 2:
Erweitern Sie das Programm jetzt noch um einen neuen, zusätzlichen Menüpunkt „**Mittelwert berechnen**“. Die entsprechende Funktion bekommt als Parameter das (gefüllte) Feld und die Anzahl der aktuell darin gespeicherten Werte. Sie gibt den berechneten Mittelwert[1] als Funktionsergebnis zurück.

[1]Der Mittelwert ist definiert als die Summe aller Elemente dividiert durch die Anzahl der Elemente

Vorbereitung (zu Hause):

- In der Headerdatei fehlen die Namen der Argumente, diese sind nämlich in C++ optional und müssen nicht angegeben werden. In den Kopfzeilen der entsprechenden cpp-Datei müssen sie natürlich angegeben werden (warum ist das so?).
- Erweitern Sie daher die Funktionsheader (auf Papier!) so, dass Sie diese direkt übernehmen können. Außerdem notieren Sie die Variablen, die Sie in den Funktionen benötigen (lokale Variablen).
- Erstellen Sie zusätzlich den passenden Eintrag für die Headerdatei der neuen Funktion für die Berechnung des Mittelwertes

Freiwillige Erweiterung:
Überlegen sie sich, wie das Menü ebenfalls als Funktion umgesetzt werden kann.

Die Headerdatei:

```
#pragma once
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

/************************************
***  Aufzaehlungstyp
************************************/
enum Diskriminante
{
      POSITIV, NEGATIV, GLEICHNULL
};

/* Hinweis: die Header-Datei verzichtet auf die Angabe der Parameternamen, da diese für
   den Übersetzungs- und Linkvorgang nicht erforderlich sind.
   In der zugehörigen *.cpp-Datei müssen diese natürlich angegeben werden.
*/

/* Die Funktion osterdatum() berechnet das Osterdatum zu einem gegebenen Jahr
   nach julianischem bzw. gregorianischem Kalender
   Eingabeparameter: Jahr
   Rückgabewert: Tageswert (Ganzzahl) des Osterdatums für März,
      Werte größer 31 liegen bereits im April und müssen im rufenden Programm
      entsprechend umgerechnet werden
   Keine Ein- oder Ausgabe in der Funktion!!
*/
int osterdatum( int );


/* Die Funktion istSchaltjahr() überprüft, ob das übergebenen Jahr (=Eingabeparameter)
   ein Schaltjahr ist oder nicht. Dementsprechend wird der Wert "true" zurückgegeben,
   wenn es sich um ein Schaltjahr handelt, sonst "false"
   Eingabeparameter: Jahr
   Rückgabewert: true oder false
   Keine Ein- oder Ausgabe in der Funktion!!
*/
bool istSchaltjahr( int );
```

```
/* Die Funktion pqformel() berechnet die reellen Lösung einer quadratischen Gleichung
   (p-q-Formel). Eingegeben werden die Koeffizienten p und q, die Rückgabe der
   Ergebnisse erfolgt über zwei weitere Referenzparameter.
   Der Rückgabewert gibt die Art der Diskriminante und damit die Anzahl der Lösungen an
   Eingabeparameter: p, q
   Rückgabewert: x1, x2 (als Parameter), Art der Diskriminante (als Rückgabewert)
   Keine Ein- oder Ausgabe in der Funktion!!
*/
Diskriminante pqformel( double, double, double &, double & );

// Die drei folgenden Funktionen behandeln die unterschiedlichen Aktivitäten für die
// Warteschlange

/* Die Funktion anfuegen() fügt ein Element am Ende der Warteschlange ein, falls dort
   noch  Platz ist.
   Als Eingabeparameter werden das Feld, das einzufügende Element (Zahlenwert), die
   aktuell vorhandene Anzahl der Elemente und die maximal mögliche Anzahl an Elementen
   erwartet.
   Als Rückgabe erhält man "true", wenn die Operation geklappt hat, sonst "false"
   Keine Ein- oder Ausgabe in der Funktion!!
*/
bool anfuegen( int [], int, int &, const int );


/* Die Funktion entfernen() löscht das erste Element der Warteschlange (und gibt es
   vorher zur Kontrolle aus)
   Als Rückgabe erhält man "true", wenn die Operation geklappt hat, sonst "false" (z.B.
   auch dann, wenn es keine Elemente zum Löschen gibt, das Feld also leer ist)
   Keine Ein- oder Ausgabe in der Funktion!!
*/
bool entfernen( int [], int & );


/* Die Funktion feldausgeben() gibt die ersten n (2. Parameter) Elemente des
   übergebenen Feldes (1. Parameter) aus.
   Keine Eingabe, aber Ausgabe in der Funktion!!
*/
void feldausgeben( int [], int );


/* Die Funktion dreiersumme() berechnet die Dreiersummen und greift dabei NUR auf
   lokale Variablen der Funktion zu. Die komplette Ein-  und Ausgabe erfolgt in der
   Funktion, es werden keine Argumente an die Funktion übergeben und auch keine
   Ergebnisse zurück geliefert.
*/
void dreiersummen();


/* Die Funktion mittelwert() berechnet den Mittelwert aller Elemente eines Feldes

   Keine Ein- oder Ausgabe in der Funktion!!
*/
```