UNIVERSITÄT BASEL

Lecturer: Thorsten Möller - **thorsten.moeller@unibas.ch**
Tutors: Nafia Nusrat - **nafia.nusrat@unibas.ch**
Sexhi Picaku - **s.picaku@stud.unibas.ch**
Mark Starzynski - **mark.starzynski@unibas.ch**

Programming Paradigms – C++ FS 2025

Exercise 1 Due: 13.04.2025 23:55:00

Upload your answers to the questions **and source code** on Adam before the deadline.

Text : For answers to questions, observations and explanations, we suggest writing them in LaTeX. Please hand-in your answers as a **single PDF** file (independent of what tools you use, LaTeX, Markdown etc.).

Source-Code : For coding exercises, the source-code must be provided and has to be **commented in detail** (e.g. how it works, how it is executed, comments on conditions to be satisfied).

Upload : Please archive multiple files into a **single compressed zip-file**, preferrably just the files without any folder hierarchy. The naming convention for your zip file is **cs109-e<ExNr>-<Name1>-<Name2>.zip**. If you upload an updated version of your solutions, the file name should contain a clear and intuitive versioning number. Only the latest version will be graded.

Requisit : In order to take the final exam, you must score at least $^2/_3$ of all available points throughout the mandatory exercises.

Modalities of work: The exercise can be completed in groups of at the most 2 people. Do not forget to provide the full name of all group members together with the submitted solution.

Question 1: Compilation Exercises (5 points)

The task of this exercise is to analyze the following C++ code snippets. Determine whether they compile successfully or result in a compilation error. If an expression does not compile, explain what the problem is and how to fix it.

1.
```
int func(int x) {}
```

2.
```
bool flag;
bool check(bool a, bool b) { return flag || b; }
check(flag, true);
```

3.
```
double compute(double val = 8.2) { return val; }
```

4.
```
typedef struct { int a, b; } Data;
```

5.
```
int main() { char c = 'A'; bool valid = isalpha(c); return 0; }
```

Question 2: Error Search (7 points)

The following files `main.cpp`, `triangle.h` and `triangle.cpp` implement a program to calculate the area of a triangle. However, the program contains errors. Identify them, explain why they are errors, and suggest fixes.

`main.cpp`

```cpp
#include "triangle.h"
#include <iostream>

using namespace std;

int main() {
    double base = base, height;

    cout << "Enter base: ";
    cin << base;

    cout >> "Enter height: ";
    cin >> height;

    TRIANGLE tri = construct_triangle(base);

    cout << "Triangle area: " << calculate(tri) << endl;

    return 0;
}
```

triangle.h

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

typedef struct {
    int base;
    double height;
} triangle;

TRIANGLE construct_triangle(int base, double height);
double calculate(TRIANGLE triangle)

#endif /* TRIANGLE_H */
```

triangle.cpp

```
#include "triangle.h"

TRIANGLE construct_triangle(double base, double height) {
    TRIANGLE tri;
    tri.base = &base
    tri.height = height;
    return triangle;
}

int calculate(TRIANGLE triangle) {
    return (1 / 2) * triangle.base * triangle.height;
}
```

The program contains several errors (there may be more if you count the same error multiple times). Find the mistakes, explain why they are mistakes, and how to fix them.

Question 3: Strings (10 points)

You are required to implement the following tasks exactly as specified, ensuring function names, parameter order, and file structure are correct to avoid point deductions.

Your project should contain the following files:

- **stringlib.h** – This header file should define a namespace called `str` and declare the required functions.
- **stringlib.cpp** – This source file should implement the functions as specified.

a) Declare and implement the function `getVowelCount` with the following properties:

- It returns an **integer value**.
- It takes a **constant reference to a string** as input.
- It counts and returns the number of vowels (`'a', 'e', 'i', 'o', 'u'`) present in the input string, case-insensitively.
- Non-alphabetic characters should be ignored.

(3 points)

b) Declare and implement the function `rotateString` with the following properties:

- It takes a **string** and an **integer** `n` as parameters and returns a **string**.
- The function should **rotate the string left by `n` positions**.
- If `n` is **greater than the length of the string**, it should wrap around.

Example:

```
Input: "hello", 2
Output: "llohe"
```

(3 points)

c) Declare and implement the function `areAnagrams` with the following properties:

- It takes **two strings** as parameters and returns a **boolean value**.
- The function determines if the two input strings are **anagrams** of each other.
- Two words are anagrams if they contain **the same characters** but in **a different order**.
- The function should be **case-sensitive**.

Example:

```
Input: "listen", "silent"
Output: true
```

(4 points)

Question 4: Structs (6 points)

a) Define a **fraction** structure (`Fraction`) with the following properties:

- A fraction consists of a numerator and a denominator.
- Implement functions for addition, subtraction, multiplication, and division.

The method prototype should look like this:

```
Fraction add(Fraction a, Fraction b);
Fraction subtract(Fraction a, Fraction b);
Fraction multiply(Fraction a, Fraction b);
Fraction divide(Fraction a, Fraction b);
```

(3 points)

b) Implement the function `simplifyFraction` that reduces a fraction to its simplest form.

```
Fraction simplifyFraction(Fraction f);
```

(2 points)

c) Write a simple test in the `main` function that shows your code works for different cases.

(1 points)

Question 5: Enums, Structs and Unions (5 points)

a) Define an **enum** called `TransactionType` with the following elements:

- `DEPOSIT`
- `WITHDRAWAL`
- `TRANSFER`

(1 points)

b) Define a **union** called `TransactionData` that contains:

- A `float` representing the amount (for deposits and withdrawals).
- A **struct** inside the union with two `float` values:
  - `amount` (transfer amount)
  - `fee` (transfer fee)

(1 points)

c) Define a **struct** called `Transaction` that contains:

- A `TransactionType` enum to represent the type of transaction.
- A `TransactionData` union to store the relevant transaction details.

(1 points)

d) Implement the function `processTransaction` that takes a `Transaction` as input and:

- Prints the amount for deposits and withdrawals.
- Prints the transfer amount and fee for transfers.

(1 points)

e) Discuss the impact of using a struct instead of a union in terms of memory usage and flexibility.

(1 points)

Question 6: Pointers (10 points)

a) In this exercise you are asked to analyze the following C++ code. Explain what is happening in each line. You do not have to run this code. What would be the output of this program?

Hint: If the output is not uniquely determinable describe of what kind it would be.

```cpp
int a = 5;
int b = 10;
int c[3] = {1, 2, 3};

int* p = &a;
int& r = b;

*p += 2;
r *= 2;

p = c;
*(p + 1) = *p + r;
p++;
*p += b - a;

cout << "a = " << a << endl;
cout << "b = " << b << endl;
cout << "c[0] = " << c[0]
        << ", c[1] = " << c[1]
        << ", c[2] = " << c[2] << endl;

p = &a;
int **q = &p;
*p *= 3**&b***q; // Never write such minified lines in your code
p = &b;
(*p)++;
cout << a << " " << b << " " << p << " " << (p + (p - 8)) << endl;
```

(4 points)

b) Write a C++ function, which takes two arrays a and b of arbitrary size. The function should return a new array, which compares element-wise array a with reversed array b and stores either the higher value in the new array, or the sum, if the values are equal. If one of the arrays is larger than the other, continue on with values from the longer array and compare them to 0. Add an option to replace the missing values of the shorter array with a default value instead.

You can use this function declaration:

```cpp
int* arrayFunction(int* a, int alen, int* b, int blen, int* deflt);
```

Example: Let's say that we get the array [1, 8, 3] and [4, 5, 3, 7, 8] as input. Then we get the array [8, 8, 6, 5, 4]; or [8, 8, 6, 10, 5] if we have chosen a default value `deflt = 5`.

(3 points)

c) In the following little program, we want to perform some calculations. But there is one but that may cause problems. Identify it, explain its consequences and propose a fix so that our code works correctly and safely again.

```cpp
#include <iostream>

int main() {
    int a = 3;
    int b = 5;
    int c[2] = {0, 0};

    int* p = &a;
    int* q = &b;
    int* r = c;

    // Store the sum of a and b in c[0]
    *r = *p + *q;
    // Store the difference of a and b in c[1]
    *(r + 1) = *p - *q;
    // Store the product of a and b in c[2]
    r[2] = *p * *q;

    std::cout << "Sum (c[0])        = " << c[0] << std::endl;
    std::cout << "Difference (c[1]) = " << c[1] << std::endl;
    std::cout << "Product (c[2])    = " << c[2] << std::endl;

    return 0;
}
```

(3 points)

Question 7: Function Pointers (7 points)

This exercise is about function pointers. In practice you often need to define an interface that can use different functions within an algorithm. Design an algorithm that can compare two arrays of the same length according to different comparator functions. More precisely, the algorithm takes two double arrays of size 3 and a comparator function as input and returns whether one array is larger than the other in regards to the given comparator. That is, return either -1, 0, 1 if the first is smaller, equal, or larger than the second one.

Implement two comparators: One that compares the two arrays according to the array sum. The other comparator considers an array to represent a point in the Euclidian space $\mathbb{R}^3$ and compares the points (arrays) according to the L2-Norm distance (see https://en.wikipedia.org/wiki/Norm_(mathematics)) from the point of origin (0, 0, 0); i.e., the first point is larger than the second if it is more distant from (0, 0, 0) regarding the L2-norm (Euclidean norm).

Use function pointers to pass the comparator function into the function.

You can use the following code skeleton to implement your functions:

pointers.cpp

```cpp
#include <iostream>
#include <math.h>

// TODO: Implement compare_sum function
// TODO: Implement compare_l2 function
// TODO: Implement compare function

int main() {
    double *array1 = new double[3] { 75, 5, 29 };
    double *array2 = new double[3] { 1, 10, 7 };
    std::cout << compare(array1, array2, compare_sum) << std::endl;
    std::cout << compare(array1, array2, compare_l2) << std::endl;
}
```

Question 8: Bonus: Shape Area Calculator (0 points)

This exercise won't be graded and thus won't get you any points, but it repeats the material from this sheet and is a good checkup for yourself to see if you have understood the concepts.

You are tasked to implement an area shape calculator. Your program should be able to instantiate objects of a type `Shape` which includes `ShapeType` (Circle, Rectangle, ...), `ShapeData` (shape attributes like radius or height for example) and a function pointer for the area calculation. After instantiating various shapes and storing them in a shape object array, your program should be able to iterate over all instantiated shapes in a for loop and output their area.

The main function of your program might look like so:

```cpp
main() {
    // allocate dynamically an array of Shape objects
    Shape* shapes = new Shape[2];

    // Initialize the first Shape as a circle
    // ...

    // Initialize the second Shape as a rectangle
    // ...

    // Print each Shape's area using the function pointer
    for (int i = 0; i < 2; ++i) {
        std::cout << "Shape " << i << " area: "
                  << shapes[i].areaFunc(&shapes[i]) << std::endl;
    }

    // What else needs to be done?
    // ...

    return 0;
}
```

What types (struct, enum, union) would you use and why? Can you improve upon the functionality of your program, for example a function that compares various properties of two shape objects?

Bonus Bonus: If you think about implementing this exact program in the Python language, what would need to change and how would this affect your program as well as your coding efforts as programmer?