UNIVERSITÄT BASEL

Lecturer: Thorsten Möller - **thorsten.moeller@unibas.ch**
Tutors: Miruna Muntean - **miruna.muntean@unibas.ch**
Mark Starzynski - **mark.starzynski@unibas.ch**

UNI BASEL

# Programming Paradigms – C++ FS 2023

## Exercise 1 Due: 16.04.2023 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**. 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)

In this exercise you should decide whether the given expression compiles or results in an error while compiling. If you think that an expression does not compile, please explain your answer. If you think an expression compiles, explain what it does.

1. `int main(int argc, char** argv) {}`

2. `float foo(int x){}`

3. `int foo(double result=10.5){return result;}`

4. `int foo(){return int z;}`

5. `typdef struct{int x,y,z;}a;`

## Question 2: Error Search (5 points)

The following three files, `main.cpp`, `bmi.h` and `bmi.cpp`, can be used to calculate a person's body mass index.

**main.cpp**

```
#import "bmi.h"

#import <iostream>

int main() {
double height;
double weight;

cout << "Enter your height (in cm): ";
cin >> height;

cout << "Enter your weight (in kg): ";
cin >> weight;

BMI bmi = construct_bmi(height, weight);

cout << "Your BMI: " << calculate(bmi) << endl;

return 0;
}
```

**bmi.h**

```
#ifndef bmi_h
#define bmi_h

typedef struct {
double height;
double weight;
} BMI;

BMI construct_bmi(double height, double weight);

float calculate(BMI bmi);

#endif /* bmi_h */
```

**bmi.cpp**

```
#include "bmi.h"

construct_bmi(double height, double weight) {
BMI bmi;
bmi.height = height / 100;
bmi.weight = weight;
return bmi;
}

double calculate(BMI bmi) {
double bmi_score = bmi.weight / (bmi.height * bmi.height);
return bmi_score;
}
```

a) The program contains 4 mistakes (more if you count the same mistake multiple times). Find the mistakes, explain why they are mistakes and how to fix them.

**(4 points)**

b) Using `g++` as a compiler, what would be the terminal command to compile the above source code into a program called `bmi`?

**(1 points)**

## Question 3: Strings, Substrings and I/O (10 points)

In this section we want you to solve two problems which include the use of the input and output of the C++ language. The user should be able to provide the inputs via the terminal and obtain the result on the terminal as well.
**Hint:** Include the *string* module by adding `#include<string>`. Useful information:

- `[i]` to access the character at position $i$
- `length()` returns the length of the string
- `insert(i,s)` inserts the string $s$ at the position $i$.

a) Here you're asked to implement a program which takes two strings $s_a$ and $s_b$. The program should "substract" the two strings by returning a new string $s_c$ in which each character of the string $s_b$ is removed from the string $s_a$ if found.
E.g.: Given the two strings $s_a$=chair and $s_b$=car, the program should output $s_c$=hi. Given $s_a$=quick and $s_b$=trick, it should output $s_c$=qu.

**(4 points)**

b) Now expand your implementation from a) with a calculation for the remainder $s_r$, the characters existing in $s_b$ and not in $s_a$.
E.g: Given two strings $s_a$=tea and $s_b$=the, the program outputs $s_c$=a and $s_r$=h.

**(2 points)**

c) Implement a function that takes a string $s_c$ and spells it backwards.
E.g.: Given the string $s_c$="abcde", the program should output $s_f$="edcba".

**(3 points)**

## Question 4: Structures in C++ (6 points)

In this exercise you should develop a function that solves the equation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

for the vector $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. Please use structs to define a `vector2D` and a `matrix2D`. The method prototype could look like this:

```
vector2D solve(vector2D a, matrix2D b);
```

## Question 5: Enum, Struct and Union (7 points)

In this section we want you to get familiar with enums, structs and unions. There is no need for user input via the terminal as in question 3. Just provide a coding example for every operand.

a) Write an enum containing an element for each of the following operands `addition`, `subtraction`, `multiplication`, `division` and `modulo`.

**(3 points)**

b) Write a function which takes an enum element from the enum you wrote in the previous exercise and two numbers and returns the result of that calculation.
E.g.: If the input enum element is `subtraction` and the numbers 5 and 1 the result should be 4.

**(3 points)**

c) Explain what a union type is in C++. How is it different to a struct containing the same data types?

**(1 points)**

## Question 6: Pointer (10 points)

a) In the following little program, we want the function `foo` to change the value of the referenced variable on which the pointer `ptr` points to. But the code is wrong on one hand and also bears a runtime problem. Explain both. What would be the correct way to do this, without changing the return type of `foo`?

```
#include <iostream>

void foo (int *ptr) {
   int *a = 5;
   *ptr = *a;
}

int main() {
   int b = 7;
   int *ptr;
   *ptr = b;
   foo(ptr);

   std::cout << *ptr << std::endl;
   return 0;
}
```

**(4 points)**

b) Write a C++ function that takes two arrays with integer elements of arbitrary size and creates a new array with the following contents. Firstly, add the reverse of the second array, keeping only the even elements. Then append the even elements of the first array, in the given order. The function returns the pointer to the beginning of the new array. Write a main function to test your implementation. You are not allowed to use square brackets to access array values. You can use this function declaration:
`int* combineReverseEven(int* a, int alen, int* b, int blen, int& len);`

Example:We get the array a $= [1, 2, 4]$ and b $= [4, 5, 6, 8, 11]$ as input. Then we get the pointer to the first element of the following array as output $[8, 6, 4, 2, 4]$.

**(3 points)**

c) In this exercise you are asked to analyze the following C++ code. What is its output? Explain what is happening in each line. Also, the last line contains the expression `(p + (p - 8))`. What is the difference to using `(p - p - 8)` instead?

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

```
int a = 4;
int b = a + 6;
int *p = &a, **q = &p;
*p *= 4**&b***q;  // Never write such minified lines in your code!
p = &b;
*p = a + 5;
(*p)++;
p *= 2;
cout << a << " " << b << " " << p << " " << (p + (p - 8)) << endl;
```

**(3 points)**

## Question 7: Function Pointers (8 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 regard to the given comparator; that is, return either of $-1, 0, 1$ if the first is smaller, equal, or larger than the second one.

Implement two comparators: One that compares the two arrays according to the sinus value of the 2nd element (see `math.h`). 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 L1-Norm distance (see `http://en.wikipedia.org/wiki/Taxicab_geometry`) 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 L1-norm.

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

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

**functionpointers.cpp**

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

// TODO: Implement compare_sin function
// TODO: Implement compare_taxicab 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_sin) << std::endl;
std::cout << compare(array1, array2, compare_taxicab) << std::endl;
}
```

## Question 8: String Analysis in Python (Optional) (0 points)

The following exercise is optional and will therefore not be graded.

You should be able to find helpful tips and example code for Python simply by searching the web, but if you don't know where to start, the official Python tutorial (`https://docs.python.org/3/tutorial/`) is a great place to get going.

Write a program in Python that:

- finds uncommon words from two Strings.
- takes a list of words and return the longest word and its length.