UNIVERSITÄT BASEL

Lecturer: Thorsten Möller - **thorsten.moeller@unibas.ch**
Tutors: Andreas Wassmer - **andreas.wassmer@unibas.ch**
Maximilian Barth - **m.barth@unibas.ch**

UNI BASEL

Programming Paradigms – C++ FS 2022

Exercise 1 Due: 19.04.2022 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)

The task of this exercise is to analyse the following code snippets. Answer which ones will compile, produce compiling errors or what they do. Also explain any errors or unexpected behaviour.

1. `int foo(int x) {}`
2. `boolean x = true;`
3. `double foo(int x) {return x;}`
4. `typedef struct a {int x, y, z;};`
5. `bool foo(int x) {return 0;}`

Question 2: Error Search (7 points)

The following three files, `main.cpp`, `pyramid.h` and `pyramid.cpp` can be used to calculate the volume of a pyramid.

main.cpp :

```cpp
#include "pyramid.h"

using namespace std;

int main() {
    int base, height;

    cout << "Enter the base side of the a sqare-based pyramid: ";
    cin << base;

    cout << "Enter the height of the pyramid: ";
    cin << height;

    PYRAMID pyramid = construct_pyramid(base, height);

    cout << "The volume of the pyramid is: " << calculate(pyramid)
         << endl;

    return 0;
}
```

pyramid.h :

```cpp
#ifndef pyramid_h
#define pyramid_h

typedef struct {
    int base;
    int height;
} PYRAMID;

PYRAMID construct_pyramid(int base, int height);

double calculate(PYRAMID &p);
```

pyramid.cpp :

```cpp
#include "pyramid.h"
#include <math.h>

PYRAMID construct(int base, int height) {
    PYRAMID p = {base, height};
    return p;
}

double calculate(PYRAMID &p) {
    double res = 1/3 * (p.base * p.base) * p.height;
    return &res;
}
```

a) The program contains 6 mistakes (there may be more if you count the same mistake multiple times). Find the mistakes, explain why they are mistakes and how to fix them.

(6 points)

b) Using `g++` as a compiler, what is the command to compile the source code above into a program called `pyramid_prog`?

(1 points)

Question 3: Strings and I/O (8 points)

In this section, we want you to solve two problems requiring the use of I/O in C++. The user should be able to provide the input and receive the output via the terminal.
Hint:

- to use strings, you need to load the *string* module with (`#include<string>`). For further info, see the string-class functions :
  https://www.cplusplus.com/reference/string/string/
- to manipulate streams, you need to load the *string-stream* module with (`#include<sstream>`)
- strings can be accessed as char-arrays (C-Strings)
- char manipulation : upper-to-lower-case by `tolower(char c)` and checking for alphabet-letters by `isalpha(char c)`.
- if inputs break up at spaces or other characters, use `getline()`

a) Here, you are asked to implement a function, which calculates the Hamming distance between two strings of equal length. The Hamming distance counts how many characters are different in each position of the given strings. For example, the Hamming distance between ”Kathrin” and ”Kerstin” is 4. The user should be able to enter the two strings into the console. Make sure to display an error message if the two strings are not of equal length (or handle unequal-size inside the distance calculation). Remember to make your program case insensitive (e.g. a = A).

b) Implement a function that prints the longest and shortest word of a given sentence, phrase or text. If two words have the same length, choose the one that occurs first.
Example: Lorem ipsum dolor sit amet. ⇒ longest = Lorem, shortest = sit
Make sure to ignore: (, . ? ! : ; or any other kind on non-letters)

(4 points)

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

In this exercise, you are asked to define your own complex number structure (`struct`). Also define four functions that perform the complex addition, subtraction, multiplication and division. The method prototype for the operations could look like this:

```
COMPLEX operation(COMPLEX a, COMPLEX b);
```

You can use the following example to test your implementation:

$$(1+5i) - \frac{(5+3i)+(2-3i)}{(2-4i)} = (0.3+3.6i)$$

Bonus Task (0 Points) : Is there a more elegant version of that implementation? Can you define such a structure in a way that you can simply write:

```
complex a,b;

cout << "addition : " << a + b << endl;
cout << "subtraction : " << a - b << endl;
cout << "multiplication : " << a * b << endl;
cout << "division : " << a / b << endl;
cout << "conuation : " << a' << endl;
```

Question 5: Enum, Struct and Union (6 points)

In this section, we want you to get familiar with enums, structs and unions.

As data, let us consider the genus "Canis" of canids [1] that contains species such as dogs and wolves. With this data, we can perform the following tests:

(There is no need for user input via the terminal as in question 3. Just code an example for every species.)

a) Write an enum called `Canids` containing an element for each of the extant species of the Canids. You can find the names of these species on the Wikipedia page. The name shoud contain the subfamily, tribe, subtribe and genus of each animal. E.g. Canis_Familiaris, Canis_Lupus

(2 points)

b) Write a function which takes an element from the enum you wrote in the previous exercise and prints the common name of the species. The method prototype should look like this:

```
void canidsCommonName(Canids c);
```

E.g.: If the input enum element is `Canis_Familiaris` the output should be `Domestic Dog`.

(2 points)

c) Explain the difference of `struct`, `union` and `class` in C++. Also provide an example how each can be used.

(2 points)

Question 6: Pointer (10 points)

a) In the following little program, we want to perform some calculations. Yet we get unexpected values or errors. Can you explain why and fix them?

Hint: Each time there only is one tiny mistake.

[1] `https://www.wikiwand.com/en/Canis`

```cpp
#include <iostream>

using namespace std;

double square(double gaga) {
    return gaga * gaga;
}

int incr(double inp) {
    int res = inp + 0.3;
    return res;
}

void reduce(double value, double result) {
    result = value - 3.1;
}

int main() {

    double a = 5, b = 6;
    int values[] = {3, 4, 7, 5, 9};

    cout << "a + b = " << *(*a) + *b << endl;

    double *c = &b;
    *c = 2;

    cout << "checking b = 2 now : b = " << *c << endl;
    cout << "geeting c^2 = " << square(c) << endl;
    cout << "increase a: " << incr(a) << endl;
    cout << "get a-th element of values: "
         << *(values + &a - 1) << endl;

    double *z = 0;
    *z = b + 1;
    cout << "z is now bigger than b: " << *z << endl;
    cout << "the 1st element of values is " << *values[0]
         << " and the second is " << ++values
         << " and the first again " << values[-1] << endl;
    {
        double c = 0;
        reduce(a, c);
        cout << "a reduced is equal " << c << 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 alternates values from a and the reversed b array. If one of the arrays is larger than the other, continue on with values from the longer array (without alternating). 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* altAntiParallel( int* a , int alen , int* b , int blen ,
                      int* deflt );
```

Example: Let's say that we get the array $[1, 2, 3]$ and $[4, 5, 6, 7, 8]$ as input. Then we get the $[1, 8, 2, 7, 3, 6, 5, 4]$ or $[1, 8, 2, 7, 3, 6, D, 5, D, 4]$ if we have chosen a default value D.

(3 points)

c) In this exercise, you are asked to analyze the following, admittedly contrived, 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.

```cpp
int a = 6;
int b = 8;
int *p = &a, **q = &p;
*p *= 3**&b***q;   /* Never write such minified lines in
                    * your code! This is just for the sake of it.
                    */
p = &b;
(*p)++;
p  = ++a;
p *= 3;
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, which can use different functions within an algorithm. Design an algorithm, which 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 it returns whether one array is larger than the other in regard to the given comparator. Returning $-1, 0, 1$ will indicate if the first array is smaller, equal, or larger than the second one.

Implement two comparators: One that compares the two arrays according to the cosine 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 [2] 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_cos 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_cos) << std::endl;
    std::cout << compare(array1, array2, compare_taxicab)
              << std::endl;
}
```

[2] `http://en.wikipedia.org/wiki/Taxicab_geometry`

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

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

Python is a popular and very versatile programming language. In a later exercise sheet there will be graded exercises that will require you to write Python programs. While this exercise is optional, it might help you prepare for future exercises.

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 [3] get started.

a) Write a function in Python that determines if

- a sentence is a palindrome. E.g.: "Taco cat" is a palindrome.
- two words are anagrams. E.g.: "listen" and "silent" are anagrams.

b) Write a function in Python that reverses a string. Add an optional parameter that causes the function to reverse the sequence of words instead of the letters.

    i.e. "I went to the future and came back to the past..."
    → "...past the to back came and future the to went I"

Note that symbols like commas and periods have to be handled like words, while apostrophs and hyphens are part of words.

[3] `https://docs.python.org/3/tutorial/`