UNIVERSITÄT BASEL

Thorsten Möller
Maarten Schenk - maarten.schenk@unibas.ch
Michael Plüss - m.pluess@unibas.ch

UNI BASEL

Programming Paradigms – C++ FS 2019

Exercise 1 Due: 31.03.2019 23:59:59

Upload answers to the questions **and source code** before the deadline via courses.cs.unibas.ch. In addition, running programs have to be demonstrated during the exercise slots until **Friday, 05.04.2019** the latest. Also note, of all mandatory exercises given throughout the course, you must score at least 2/3 of the total sum of their points to be allowed to take the final exam.

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 (6 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. `float foo(int y){}`

2. `int foo(){return int x;}`

3. `void foo(int z){return z;}`

4. `typedef struct{int x,y,z;}a;`

5. `int foo(double q=5.5){return q;}`

6. `static int s;`

Question 2: Error Search (5 points)

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

main.cpp

```
import "cuboid.h"
import <iostream>

use namespace std;

int main() {
  double height, width, length;

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

  cout << "Enter cuboid width: ";
  cin >> width;

  cout << "Enter cuboid length: ";
  cin >> length;

  CUBOID cuboid = new CUBOID(height, width, length);

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

  return 0;
}
```

cuboid.h

```
#define cuboid_h
#ifndef cuboid_h

typedef struct {
  double height;
  double width;
  double length;
} CUBOID;

CUBOID construct_cuboid(double height, double width, double length);

double calculate(CUBOID cuboid);

#endif /* cuboid_h */
```

cuboid.cpp

```
#include "cuboid.h"

CUBOID construct_cuboid(double height, double width, double length) {
  cuboid.height = height;
  cuboid.width = width;
  cuboid.length = length;
  return cuboid;
}

double calculate(CUBOID cuboid) {
  double cuboid_volume = cuboid.height * cuboid.width * cuboid.length;
}
```

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.

(4 points)

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

(1 points)

Question 3: Strings and Substrings (8 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: You can include the *string* module by adding `#include<string>` to the top of your file. The following functions and operators might be useful

- `[i]` accessing 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=pasta and s_b=pizza, it should output s_c=sta.

(4 points)

b) Now expand your implementation from a) with a calculation for the remainder s_r. This is equal to asking, which additional characters exist in s_b compared to s_a.
E.g.: Given the two strings s_a=tea and s_b=tree, the program should output s_c=a and s_r=re.

(1 points)

c) Implement a function that takes a string s_c and transforms lower case letter to upper case letters and vice versa.
E.g.: Given the string s_c="AbcDe", the program should output s_f="aBCdE".

(3 points)

Question 4: Structures in C++ (7 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 code an example for every operand.

a) Write an enum called operand 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 `addition` and the numbers 4 and 5 the result should be 9.

(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: Plotter **(12 points)**

In this exercise you're asked to implement a simple version of a plotter for a polynomial function. This plotter should be approximately able to mark extrema of the function by calculating the local derivative. The local derivative of a function is defined as follows:

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

The function you should plot is:

$$f(x) = x^3 - x$$

Since the quality of the output greatly depends on the input parameters you're not asked to deliver an exact result. A sample output could look like this:

```
                                    |                     #
                                    |
                                    |
                                    |                    #
                                    |
                                    |                   #
                         #^^#       |
                       ##    ###    |                  #
                     ##         ##  |                 #
                    #             ##|
------------------#-----------------#----------------#-----------------
                                    |##             #
                 #                  |  ##         ##
                #                   |    ###    ##
                                    |       #vv#
               #                    |
                                    |
              #                     |
                                    |
                                    |
```

Together with the exercise, you can find a skeleton file *plotter.cpp* which you don't need to use if you don't want to, but which may help you get started.

Question 7: 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 (`https://docs.python.org/3/tutorial/`) is a great place to get going.

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.