UNIVERSITÄT BASEL

Lecturer: Xxxxxxxx Xxxxxxx - **xxxxxxxx.xxxxxxx@unibas.ch**
Tutors:
Xxxxxxxxx Xxxxxxxxxxx - **xxxxxxxxx.xxxxxxxxxxx@stud.unibas.ch**
Emanuele Tirendi - **emanuele.tirendi@unibas.ch**
Xxxx Xxxxxxxx - **xxxx.xxxxxxxx@unibas.ch**

Programming Paradigms – C++ FS 2024

Exercise 1 Due: 21.04.2024 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 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 are correct and which ones are not correct i.e. which ones will produce compiler or linker errors/warnings. For the ones that are not correct, explain what the problem is and how to fix it.

1. `double foo(double x) {}`
2. `bool x; bool foo(bool y, bool z) { return x && z;} foo(x, x);`
3. `int foo(int x) {return x;}`
4. `typedef enum {int x, y, z;}a;`
5. `int main(int argc, char** argv) {bool b = foo(3); return 0 }`
   `bool foo(int x) { return 0 && x;}`

Question 2: Error Search (7 points)

The files `lib.h` and `lib.cpp` could represent a library which provides as its functionality the structs `libstruct1` and `libstruct2` and the function `libfunctionality`.

main.cpp :

```
1  #include "lib.h"
2  #include <iostream>
3
4  using namespace std;
5
6  int main(int argc, char** argv) {
7      int z;
8      cin << z;
9      int i = libfunctionality(z, true);
10     cout << i;
11     return 0;
12 }
```

lib.h :

```
1  #ifndef LIB_H
2  #define LIB_H
3  #include "lib.cpp"
4
5  namespace libnmsp {
6      struct libstruct2;
7
8      struct libstruct1 {
9          int i;
10         bool b;
11         char c;
12         libstruct2 l2;
13     }
14
15     struct libstruct2 {
16         int j;
17         libstruct1 l1;
18     }
19
20     int libfunctionality(int i, bool b);
21 }
22
23 #endif
```

lib.cpp :

```
#include "lib.h"

int libnmsp::libfunctionality(int i, bool b) {
    int j = i + getNumber(b);
    std::cout << j << "\n";
}

int getNumber(bool b) {
    if (b == true) {
        return 1;
    } else {
        return 0;
    }
}
```

The program contains 7 compiler/linker errors/warnings. (It contains more than 7 if you count the same error multiple times.) Find the mistakes, explain why they are mistakes and how to fix them.

Question 3: Arrays (13 points)

In this section, we want you to implement a header file (a file with the extension .h) and an implementation file (a file with the extension .cpp) for functionality that can be used for the manipulation of 0-terminated character strings.

Note that you must not use any library for this task and should implement everything on your own.

Implement everything exactly as described below, even the order of parameters the functions take or the name of the functions, files or namespaces. Doing otherwise may result in points being deducted.

a) We want you to write two files, `arraylib.h` and `arraylib.cpp`. In `arraylib.h` we want you to define your own namespace called `arr`. Declare within the namespace in `arraylib.h` and implement in `arraylib.cpp` the function `getLength` which has the following properties:

- It returns an integer value.
- It takes a character pointer which is interpreted as a pointer to the first element of a 0-terminated character string.
- If we have a character string `char c[] = "hello"` the character string `c` has length 6, because the last character is the character `'\0'` which is implicitly added by the compiler. We want `getLength` to return 5 as it is the length seen in the definition of this array.
- If the length of the array (the number of the characters stored in the array without the terminating 0-character) is bigger than 20, getLength should return 0.

(3 points)

b) We want you to declare within the namespace `arr` in `arraylib.h` and implement in `arraylib.cpp` the function `replaceAllSubarrays` as follows:

- It returns a boolean value
- It takes three character pointers in the following order: `array`, `toBeReplaced` and `toReplaceWith`.
- The three character pointers are interpreted as pointers to the first element of 0-terminated character arrays. If the length of `toReplaceWith` and `toBeReplaced` aren't the same, nothing happens and `replaceAllSubarrays` returns the boolean value `false`.
- If the length of `array` is smaller or equal to the length of either `toReplaceWith` or `toBeReplaced`, nothing happens and `replaceAllSubarrays` returns the boolean `false`.
- If the length of one of the three input arrays without the terminating 0-character

is bigger than 20 or euqal to 0, nothing happens and `replaceAllSubarrays` returns the boolean value `false`.

- Else, `replaceAllSubarrays` returns the value `true` and replaces every occurence of `toBeReplaced` in `array` by `toReplaceWith`. E.g. if we have `array="xhalloahallowww"`, `toBeReplaced="hallo"` and `toReplaceWith="world"` then after calling the function `replaceAllSubarrays` the array `array` should be equal to `"xworldaworldwww"`. Note that every character of `array` can be part of only one occurrence, namely the one starting at the smallest index. If we have for example `array="xxxababababxxx"`, `toBeReplaced="abab"` and `toReplaceWith="cdcd"`, then `replaceAllSubarrays` should change `array` correctly to `"xxxcdcdabxxx"` and **not** to `"xxxcdcdcdxxx"` or `"xxxabcdcdxxx"`.

(6 points)

c) We want you to declare in within the namespace `arr` in `arraylib.h` and implement in `arraylib.cpp` the function `intersection` as described below. We chose this name since the operations this function will perform on the arrays has similarities with the set operation of the same name.

- It returns a boolean value.
- It takes three character pointers in the following order: `a`, `b` and `intersection`.
- The three character pointers are interpreted as pointers to the first element of 0-terminated character arrays. If the length `a` and `b` aren't the same, nothing happens and the function `intersection` returns the value `false`.
- If the length of the array `intersection` isn't double as big as the length of `a` or `b`, nothing happens and the function `intersection` returns the value `false`.
- If the length of `a` and the length of `b` both without the terminating character are bigger than 10 or are equal to 0 and if the length of the array `intersection` without the terminating character is bigger than 20 or equal to 0, nothing happens and the function `intersection` returns the value `false`.
- Let's denote with $s_{a|b}$ the string containing all the characters of `a` that are also present in `b` in the same order as they occur in `a`. That means, that if `a="hallo` and `b="olten"` then $s_{a|b} =$ "llo" and $s_{b|a} =$ "ol". Let further be $s_1 \frown s_2$ the concatenation of two strings, meaning if $s_1 =$ "$hello$" and $s_2 =$ "$world$", then $s_1 \frown s_2 =$ "$helloworld$.

  If all other cases described above can be excluded, we want the function `intersection` to return true and to calculate $s_{a|b} \frown s_{b|a}$ and store it in the array `intersection`. Since $s_{a|b} \frown s_{b|a}$ usually has length smaller equal to the length of the array `intersection`, we want the function `intersection` to store $s_{a|b} \frown s_{b|a}$ starting at index 0 of the array `intersection`. Every index of the array `intersection` that won't be overwritten by $s_{a|b} \frown s_{b|a}$ should remain as it was before calling the function `intersection`.

(4 points)

Question 4: Structs (4 points)

a) In this exercise, you'll develop a function `getInverse` that takes a $2x2$-matrix and returns the inverse of that matrix if it is defined. It returns a matrix with all entries equal to 0 if the inverse is not defined. If you find an explicit representation of the inverse of a $2x2$-Matrix on the internet, this task will be easy to solve. Please use structs to define a `matrix2D`. The method prototype could look like this:

```
matrix2D getInverse(matrix2D matrix);
```

Write a simple test in the main function that shows that your code works with both, an invertible matrix as well as a matrix that cannot be inverted. Not writing tests may result in points being deducted.

(2 points)

b) In this exercise, you'll develop a function `solveEquation` that takes a $2x2$-matrix M and a vector with two elements v and returns the solution x of the equation $Mx = v$. If the matrix is not invertible, the solution should have only 0 entries. If you use the function defined in a) this task will be easy to solve. Please use structs to define a `vector2D`. The method prototype could look like this:

```
vector2D solveEquation(matrix2D m, vector2d v);
```

Write a simple test in the main function that shows that your code works with both, an invertible matrix as well as a matrix that cannot be inverted. Not writing tests may result in points being deducted.

(2 points)

Question 5: Enums, Structs and Unions (8 points)

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

a) Write a single enum containing an element for each of the following arithmetic operations: addition, subtraction, multiplication, division and an element for each of the following logical operations: and, or, nand, xor. Write also an union containing an integer and a boolean.

(2 points)

b) Write a function which takes an enum element from the enum you have written in the previous exercise and two instances of the union you have written. Provide an example for every operand in your main function. Make sure to initialize the unions correctly so that the initialization matches the operations used.

- If the enum provided is an arithmetic operation, the function should return a union containing an integer that is the result of the operation provided by the operation appied to the two integers of the union instances the function has got as arguments. E.g. if the unions name is `BoolInt` and we have `BoolInt a, c; a.i = 2; c.i = 3;` where `i` is the name of the integer in `BoolInt` and the function's name is `arithLogic`, then the following code should print a 5: `BoolInt result = arithLogic(a, c, ADDITION); std::cout << result.i;`
- If the enum provided is a logic operation, the function should return a union containing a boolean that is the result of the operation provided by the operation applied to the two booleans of the union instances the function has got as arguments. E.g. if the unions name is `BoolInt` and we have `BoolInt a, c; a.b = true; c.b = false;` where `b` is the name of the boolean in `BoolInt` and the function's name is `arithLogic`, then the following code should print a 0: `BoolInt result = arithLogic(a, c, AND); std::cout << result.b;`. Note that booleans printed in std::cout are interpreted as numbers ($1 = true$, $0 = false$).

(4 points)

c) Would it be possible to change the function in a way that it takes two enums, one representing an arithmetic operation and one representing a logic operation, and return an instance of the union that has both the integer and the boolean as the correct result of the operations applied to the integers and booleans of the union instances taken as arguments? Explain your answer.

(1 points)

d) Would it be possible with structs instead of unions? Explain your answer.

(1 points)

Question 6: Pointers (5 points)

In this exercise you are asked to analyze the following C++ code. What is its output? Explain what is happening in each line

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

functionpointers.cpp

```
1
2  #include <iostream>
3
4  int main() {
5          int a = 3;
6          int b = a*a;
7          int c[] = {1, 3, 5, 7};
8          int *p = &a, **q = &p, *o = &b;
9          *p = *o;
10         *c += *(c + 2);
11         *(c + 3) *= *c + 2;
12         *p = 4**&b***q;
13         (**q)++;
14         std::cout << p << " " << (*p + (*p - 8)) << std::endl;
15         *o *= 2;
16         std::cout << a << " " << p << " " << ((c + 1)[0]) << "\n";
17         std::cout << c[0] << " " << c[1] << " " << " " << c[3];
18 }
```

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 takes two integer arrays of size 3 which serve as operands, a boolean array which serves as result of the operation, and an operator-function. This operator-function itself takes two integer arrays of size 3 and a boolean array.

Implement two operator-functions:

- One that stores at index `i` of the boolean value the value `true` if the number stored at the same index of the first operand-array is greater-equal to the number stored at the same index of the second operand-array, and the value `false` otherwise.
- The second operator-function stores at index `i` of the boolean array if the product of the numbers of the operand-arrays stored at the same index is greater-equal to zero or not.

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

You can use the following code skeleton to implement your functions. It is mandatory to provide test to show that your code works.

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

// TODO: Implement greaterEqual function
// TODO: Implement productNonNegative function
// TODO: Implement operation

int main() {
    int *array1 = { 75, 5, 7 };
    int *array2 = { -1, 10, 7 };

    bool *result1 = {0, 0, 0};
    operation(array1, array2, result1, greaterEqual);
    bool *result2 = {0, 0, 0};
    operation(array1, array2, result2, productNonNegative);

    for (int i = 0; i < 3; i++) {
        std::cout << result1[i] << " ";
    }
    std::cout << "\n";

    for (int i = 0; i < 3; i++) {
        std::cout << result2[i] << " ";
    }
    std::cout << "\n";
}
```

Question 8: Bonus: Theory of Programming Languages (0 points)

This exercise won't be graded and thus won't get you any points, but you're recommended to solved it since the content of this exercise (i.e. the introductory part of the course: theory of programming languages) is relevant for the exam. So, use this exercise as an opportunity to check if you've understood the concepts of the introductory part and as an opportunity to let your answers be corrected by the tutors.

a) You've learned that type systems can be classified into three dimensions where each dimension can manifest itself in two forms.

   Provide for each form of each dimension an example where the type system has this form in this dimension and give a brief explanation why this is so. The programming languages you can use are C++, Java and Python.

b) Consider the following pseudo code. What will be the output when this code is executed with both lexical scope and dynamic scope?

```
1  n = 100
2
3  def clam()
4      display "in clam, n="
5      display n
6      display newline
7
8  def squid(n)
9      display "in squid, n="
10     display n
11     display newline
12     clam()
13     n = 5
14
15 display "in main program, n="
16 display n
17 display newline
18 squid(1)
19 clam
```