

Midterm Self-Evaluation FS2020

C++ Programming I
09 April 2020

Name: Stefan Weber	Exam-ID: 1
--------------------	------------

Question	1	2	3	4	5	6	7	8	9	10	Σ
Scored points											
Maximum points	3	4	2	2	3	4	4	4	6	6	38

Information:

This is a **self-evaluation test**, very similar to the midterm exams of the past years! The goal is to achieve at least **50%** of the overall points, *i.e.* 19 points, to pass and to get a regular exercise point.

- ► The duration of the exam is 60 minutes determined by ilias!
- ▶ Please use the provided empty **CMake-project as a template** for your solution. Modify and extend as required.
- ► The exam is open book!
- ▶ Work on your own!
- Justify your answers.
- ▶ Have fun!

Question 1: General (3 Points)

Write a program helloWorld that prints out a "Hello World" message. An example output might look like this: ./helloWorld Hello World!

Your code:

```
see q1 file

see q1 file

//
```

Question 2: Functions (4 Points)

You're given the following code below. Implement the corresponding functions to calculate the maximum and average value of the array.

```
#include <iostream>
     // Add your functions here
10
11
12
                  see q2 file
16
17
18
19
20
21
22
23
24
25
28
29
30
31
32
33
34
35
36
37
38
     int main ()
40
          int len = 10;
          double data[] = {2.4, 5.2, 3.7, 1.9, 7.4, 3.4, 4.6, 3.9, 6.4, 5.8};
41
42
          double maximum = maxValue(data, len);
43
          double average = averageValue(data, len);
45
          std::cout << "Maximum: " << maximum << std::endl;
std::cout << "Average: " << average << std::endl;</pre>
48
          return 0;
49
```

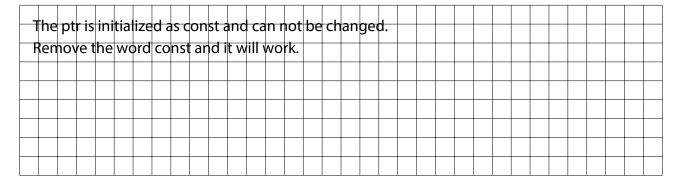
Exam-ID: 1

Question 3: Pointers & References (2 Points)

What's wrong with the following snippet? Please explain and fix.

```
int main()
{
    int x = 5;
    int y = 7;

    const_int* ptr = &x;
    std::cout << *ptr;
    *ptr = 6;
    std::cout << *ptr;
    ptr = &y;
    std::cout << *ptr;
    ptr = std::cout << *ptr;
    ptr = std::cout << *ptr;
    ptr = o;
    std::cout << *ptr;
    std::cout << *ptr;
```



Question 4: Arrays (2 Points)

What's wrong with the following snippet? Please explain and fix.

```
int* allocateArray(const int length)
{
    int temp[length];
    return temp;
}
int *temp = new int[length];
}
```

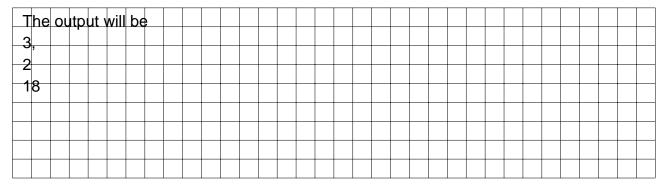
For dynamical allocation of memeory the word now must be used. Compiler doesn't know how much memory should be reversed because length is not known.

Question 5: Pointers & References (3 Points)

What is the output of the following program?

```
#include <iostream>
int h(int v)
{
    return 3*v;
}

int g(int *v)
{
    return *v *= 2;
}
```



Question 6: Classes (4 Points)

A color is given by its 3 components R,G and B. Implement a class Color (header-only), which allows the component-wise addition of two colors, *i.e.* additive color mixing in C++ as follows:

```
Color x(1,0,0), y(0,1,1);
Color z = x.add(y); // z = (1,1,1) Solution also in color.h
```

Your header only implementation in color.h

```
#ifndef COLOR_H
   #define COLOR_H
   class Color
       protected:
       int m_Red, m_Green, m_Blue =0;
10
       public:
         Color(int red, int green, int blue):m_Red(red), m_Green(green), m_Blue(blue){};
14
15
16
17
         Color add(Color y){
19
20
            return Color(y.m_Red+m_Red, y.m_Green+m_Green,y.m_Blue+m_Blue);
21
22
         }
26
27
28
29
31
32
33
34
35
39
   #endif // COLOR_H
```

Question 7: Inheritance (4 Points)

Extend class Color (RGB) to ColorA (RGBA) by a fourth opacity component, the so called alpha channel, modelling the transparency. To composite a colorA C obtained by adding ColorA A with opacity α_A over colorA B with opacity α_B , alpha-blending is used:

$$C = (\alpha_B B + (1 - \alpha_B)\alpha_A A) \frac{1}{\alpha_C} \quad with \quad \alpha_C = \alpha_A + (1 - \alpha_A)\alpha_B$$

```
ColorA a(1,0,0,0.5), b(0,0,1,0.5); ColorA c = a.add(b); // c = (1/3, 0, 2/3, 3/4) Solution also in colora.h
```

Extend and add your header only implementation of class ColorA with member function add below:

```
#ifndef COLORA_H
    #define COLORA_H
    class ColorA:public Color
                   private:
                   double m_alpha = 0;
10
                   public:
11
                   ColorA(double red, double green, double blue, double alpha):Color(red, green, blue), m_alpha(alpha){};
13
                   ColorA add(ColorA b){
16
                   double m_alphaC = m_alpha + (1 - m_alpha)*b.m_alpha;
17
18
19
                 return ColorA((b.m_alpha * b.m_Red + (1-b.m_alpha) * m_alpha * m_Red) * (1/m_alphaC), (b.m_alpha *
20
                 b.m_Green + (1-b.m_alpha) * m_alpha * m_Green) * (1/m_alphaC), (b.m_alpha * b.m_Blue + (1-b.m_alpha)
22
                 b.m_alpha) * m_alpha * m_Blue) * (1/m_alphaC), m_alphaC);
23
24
25
                 };
26
29
30
31
32
35
36
37
38
    #endif // COLORA H
```

Question 8: Inheritance - Access Specifier (4 Points)

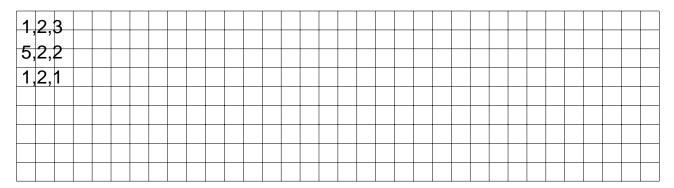
Below you find an example of public, protected and private inheritance. Are Base's public member variables accessible in the derived $(D1_*)$ and further derived $(D2_*)$ classes? If not, which modifications are necessary? Mark the correct answer(s).

- (1) Which of the further derived classes (D2_*) have access to m_a and m_b?
 - ber variables.
 - (A) D2_priv has access to Base's public mem- (C) D2_pub has access to Base's public member variables.
 - (B) D2_prot has access to Base's public member variables.
- (D) All have access.
- (2) Which classes need modifications to allow access to Base's public member variables?
 - (A) D1_priv needs to be modified.
- (C) D1_publ needs to be modified.
- (B) D1_prot needs to be modified.
- (D) None.
- (3) Modify at least one derived class, to allow access to Base's public member variables. class D1_priv: public Base{ /* . . . */ };

Question 9: Initialisation (6 Points)

What is the output of the program shown below?

```
#include <iostream>
       class A
       public:
               int m_v = 5;
                int m_u= 2;
               int m_w;
                \begin{array}{l} A() \; : \; m\_v\{1\}, \; m\_w\{2\} \; \{ m\_u++; \} \\ A(\mbox{const} \; A \; \&a) \; : \; m\_w\{a.m\_w\} \; \{ \} \\ \end{array} 
                A(A \&\&a) : m_v{a.m_v}, m_w{m_u} {--m_u;}
13
       int main ()
14
15
16
                A v2(v1);
                A v3(std::move(v1));
               std::cout << v1.m_v << ',' << v1.m_w << ',' << v1.m_u << std::endl;
std::cout << v2.m_v << ',' << v2.m_w << ',' << v2.m_u << std::endl;
std::cout << v3.m_v << ',' << v3.m_w << ',' << v3.m_u << std::endl;
20
21
                return 0;
```



Question 10: Memory Management (8 Points)

You're given the implementation of a bitSet below, i.e. a class holding an array of boolean values. Unfortunately the implementation of bitSet leaks memory and shows undefined behaviour:

```
#include <iostream>
#include "bitset.h"

void useBitSet(BitSet bSet){/* Doing something smart with bitset */}

int main()
{
    BitSet bSet(10);
    useBitSet(bSet);
    BitSet bset2 = std::move(bSet);
    return 0;
}
```

```
Please fix the issues by writing additional or extending existing code, but without deleting code!
    #define BITSET_H
    class BitSet
   private:
6
7
        bool* m_bitSet;
        int m_nbrValues;
10
        BitSet(int nbrValues) : m_nbrValues(nbrValues), m_bitSet( new bool[m_nbrValues] )
11
12
           /* CTor Code */
13
        }
14
        ~BitSet()
        {
            delete m_bitSet;
18
19
20
21
        // You can add your code below
24
              See bitset.h
25
26
27
29
30
31
32
33
35
36
37
38
39
41
42
43
44
45
46
48
49
50
51
55
56
```

#endif // BITSET_H