Bern University
of Applied Sciences

394661-FS2018-0 - C++ Programming I

EXERCISE-03

TABLE OF CONTENTS

1 Introduction

This exercise of 394661-FS2018-0 will focus on pointers and references in C++. The goal of this exercise is to point out the difficulties and pitfalls when working with raw memory and, hence, pointers. To make life easier you'll get in contact with one of the so called smart-pointers, i.e. the unique pointer, which handles memory management for you. Finally, as always recommended, you'll use a dynamic standard container and some standard algorithms to implement the problems.

You will learn the following topics when completing this exercise:

- Pointers
- Dynamic memory
- Memory leaks
- Array and pointers
- First smart pointer
- Standard vector and use of some algorithms

2 Excercises

2.1 Allocating and Freeing Memory

In this exercises you'll write a simple console program asking the user for a number of integer values to enter, summing them up and showing the result to the user. In addition, the minimum and maximum value of the entered data is evaluated, *e.g.*:

```
How many values do you want to enter?: 5
1. value: 2
2. value: 1
3. value: 24
4. value: 12
5. value: 18
---------------

Results:
Sum: 57
Min: 2
Max: 24
```

Implement the program using:

1. "unsafe" **arrays**, *i.e.* allocate arrays dynamically with `new`, `delete` and
   a) calculate results with **index access** using the operator `[]`
   b) calculate results with **pointer arithmetic** using `*`
2. the safe C++11 **smart pointer** type `std::unique_ptr` (`#include <memory>`):

```cpp
// Include header (C++11!)
#include <memory>

// Create unique pointer to int array
std::unique_ptr<int[]> smartPtr(new int[size]);

// Access array normally
int value = smartPtr[i];
```

3. the very safe and convenient **STL-container** `std::vector` (`#include <vector>`) and STL-algorithms `std::accumulate`, `std::min_element` (`#include <algorithm>`)
   `http://www.cplusplus.com/reference/algorithm/min_element/`

2.2 Pitfalls

Comment and fix the problems!

- What's wrong here?

```cpp
#include <iostream>

int main()
{
    int *pointToAnInt = new int;
    pointToAnInt = 9;
    std::cout << "The value at pointToAnInt: " << *pointToAnInt;
    delete pointToAnInt;
    return 0;
}
```

- Why and where does the program crash?

```cpp
#include <iostream>

int main()
{
    int pointToAnInt = new int;
    int* pNumberCopy = pointToAnInt;
    *pNumberCopy = 30;
    std::cout << *pointToAnInt;
    delete pNumberCopy;
    delete pointToAnInt;
    return 0;
}
```

- Fix the function!

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

- Hoppla!

```cpp
int main()
{
    int array[5] { 0, 1, 2, 3 };
    for (int count = 0; count <= 5; ++count)
        std::cout << array[count] << " ";

    return 0;
}
```

3 Submission

Submit your source code (as a zip-file) to Ilias Ex-03 **before the deadline** specified in Ilias.