Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecturer: Prof. B. Gärtner

Informatik für Mathematiker und Physiker HS15

Exercise Sheet 9

Submission deadline: 15:15 - Tuesday 17th November, 2015
Course URL: http://lec.inf.ethz.ch/ifmp/2015/

Short Summary

	Array	Vector
Array/Vector Initialization	`int my_arr[] = {1,2,3};`	`std::vector<int> my_vec = {1,2,3}; // 1 2 3` `std::vector<int> my_vec (3, 1); // 1 1 1`
Iterator to Begin	`int*  ptr  = my_arr;`	`std::vector<int>::iterator  itr  = my_vec.begin();`
const	`const int* cptr = my_arr;`	`std::vector<int>::const_iterator citr = my_vec.begin();`
Iterator to Past-the-End	`int*  ptr  = my_arr + 3;`	`std::vector<int>::iterator  itr  = my_vec.end();`
const	`const int* cptr = my_arr + 3;`	`std::vector<int>::const_iterator citr = my_vec.end();`
to Next Elt	`++ptr`	`++itr`
to Previous Elt	`--ptr`	`--itr`
Distance	`ptr1 - ptr2`	`itr1 - itr2`
Comparisons	`ptr1 < ptr2`	`itr1 < itr2`
	`ptr1 != ptr2`	`itr1 != itr2`
	…	…

Assignment 1 (8 points)

This exercise consists of 4 independently solvable subtasks. **For the whole exercise make sure that your codes are `const`-correct.** The following `typedef`s are used to simplify the notation:

```
typedef std::vector<int>::iterator Vit;
typedef std::vector<int>::const_iterator Cvit;
```

a) Write a function

```
// PRE: [begin, end) is a valid non-empty range.
// POST: returns the average of the elements in [begin, end)
double average (const double* begin, const double* end);
```

to compute the average of the elements in an array with elements of type `double`. You can use the template `ex1a_template.cpp` to submit to the judge.

Judge Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2015/judge_boxes.html)

```
Input 20 numbers: 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10
Average = 5.5
```

Submission: https://challenge.inf.ethz.ch/team/websubmit.php?cid=5&problem=MP15091a

b) Write a function

```
// PRE: [begin, end) is a valid non-empty range
// POST: returns the value of the largest element in [begin, end)
int find_max (Cvit begin, Cvit end);
```

to find the value of the largest element in a vector with elements of type `int`. You can use the template `ex1b_template.cpp` to submit to the judge.

Judge Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2015/judge_boxes.html)

```
Length =? 4
Input numbers: 1 -1 2 0
Maximum = 2
```

Submission: https://challenge.inf.ethz.ch/team/websubmit.php?cid=5&problem=MP15091b

c) Write a function

```
// PRE: [begin, end) is a valid range
// POST: fills the range with the following pattern
//         1 2 2 3 3 3 4 4 4 4 ...
//       The pattern is cut off as soon as the range is full.
void n_times_n_pattern (Vit begin, Vit end);
```

to fill the range with the specified pattern. You can use the template `ex1c_template.cpp` to submit to the judge.

Judge Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2015/judge_boxes.html)

```
Length =? 16
1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6
```

Submission: https://challenge.inf.ethz.ch/team/websubmit.php?cid=5&problem=MP15091c

d) Write a function

```
// PRE: [begin, end) is a valid range
// POST: returns the length of the longest non-decreasing subrange
//       in [begin, end)
unsigned int longest_nondecr_subrange (const int* begin, const int* end);
```

to determine the length of the longest sequence of consecutive elements which are non-decreasing. You can use the template `ex1d_template.cpp` to submit to the judge.

Judge Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2015/judge_boxes.html)

```
Input 20 numbers: 1 2 3 1 2 3 1 2 3 4 1 2 3 4 5 6 7 8 9 10
Length of longest non-decreasing subrange: 10
```

Submission: https://challenge.inf.ethz.ch/team/websubmit.php?cid=5&problem=MP15091d

Assignment 2 (4 points)

Which of the following keywords stand in relation with each other? State the most important relations and explain each relation in your own words. Notice that there might be more than one relation between two keywords. Find and explain at least 10 relations, and make sure that you mention each keyword at least once.

Pointer typedef Array Vector Iterator Container Function Subscript Operator []

Two relations as an example:

- **Vector** --> can do more than --> **Array**
- **Iterators** --> are used to pass **Containers** to --> **Functions**

Assignment 3 – Skript-Aufgabe 126 (4 points)

In how many ways can you own CHF 1? Despite its somewhat philosophical appearance, the question is a mathematical one. Given some amount of money, in how many ways can you partition it using the available denominations (bank notes and coins)? Today's denominations in CHF are 1000, 200, 100, 50, 20, 10 (banknotes), 5, 2, 1, 0.50, 0.20, 0.10, 0.05 (coins). The amount of CHF 0.20, for example, can be owned in four ways (to get integers, let's switch to centimes): $(20), (10, 10), (10, 5, 5), (5, 5, 5, 5)$. The amount of CHF 0.04 can be owned in no way, while there is exactly one way to own CHF 0.00 (you cannot have 4 centimes in your wallet, but you *can* have no money at all in your wallet).

Solve the problem for a given input amount, by writing the following function (all values to be understood as centimes).

```
// PRE: [begin, end) is a valid nonempty range that describes
//      a sequence of denominations d_1 > d_2 > ... > d_n > 0
// POST: return value is the number of ways to partition amount
//       using denominations from d_1, ..., d_n
```

```
unsigned int partitions (unsigned int amount,
                         const unsigned int* begin,
                         const unsigned int* end);
```

On the course webpage you find the program `partitions_template.cpp` where you can plug-in your code and use it to test your function and to submit to the judge.

Judge Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2015/judge_boxes.html)

```
In how many ways can I own x CHF-centimes for x =? 0
1
```

```
In how many ways can I own x CHF-centimes for x =? 95
39
```

```
In how many ways can I own x CHF-centimes for x =? 555
8825
```

Submission: https://challenge.inf.ethz.ch/team/websubmit.php?cid=5&problem=MP15093

Challenge – (8 points)

It may be interesting to know in how many ways you can own CHF 1 (Assignment 3), but wouldn't it be much more satisfactory to talk about CHF 100, or even CHF 1000? Unfortunately, with the standard solution of Assignment 3 you won't get there. This challenge is about getting the answer also for larger amounts of money. And to enjoy the endless possibilities that come with such larger amounts.