technische universität dortmund

Department of Computer Science
Chair 11 / Algorithm Engineering

Dr. Carsten Gutwenger

Winter 2011/12

Object-oriented Programming
Assignment Sheet No. 3

Date: November 8

Exercise 3.1 (Strings)

Write a program that reads a sequence of words from the console until the word `stop` is entered, and then prints the reversed sequence (without the final `stop`).

Example:

```
Input:  Gallia est omnis divisa in partes tres stop
Output: tres partes in divisa omnis est Gallia
```

Exercise 3.2 (Vectors and Floats)

Write a program that reads a sequence of floating-point values (type `float`) from the console until the user enters 0, and then prints the sequence (without the final 0) in ascending order without duplicates.

Example:

```
Input:  1.5 32 -2 1.5 2 2 55 32 1.5 0
Output: -2 1.5 2 32 55
```

Exercise 3.3 (Vectors and Formatted Output)

Write a program that reads a sequence of positive integers (until the user enters a non-positive number) and then prints a histogram of all the integer value between the smallest and the largest integer the user has entered, i.e., for each such value the number of occurrences of that value in the sequence entered by the user.

Example:

```
Input:  5 10 12 7 10 10 5 11 7 12 10 0
Output:

value | occurrences
-------------------
    5 | 2
    6 | 0
    7 | 2
    8 | 0
    9 | 0
   10 | 4
   11 | 1
   12 | 2
```

Exercise 3.4 (Vectors)

Remark: This exercise is challenging and goes beyond tasks you will be given in the exam sheets.

Write a program that prints all the prime numbers smaller or equal to a number n chosen by the user. Recall that a *prime number* is a natural number $p > 1$ that has no other positive integer divisors other than 1 and p itself.

Use the *sieve of Eratosthenes* algorithm to compute the prime numbers: It starts with a list of numbers $2, \ldots, n$ which are candidates for prime numbers. Each of these numbers can be marked or unmarked; initially all are unmarked. Then start with the smallest unmarked number p in the list (which is initially 2). This number is the next prime number we can print. Then mark all multiples of p in the list (they cannot be a prime number anymore) and find again the smallest unmarked number in the list, which is the next prime number. Proceed until no more unmarked numbers are in the list.

Example: Let $n = 10$.

1. We print 2 and mark $2, 4, 6, 8, 10$.
2. Now the list of unmarked numbers is $3, 5, 7, 9$, so we choose 3 and mark $3, 6, 9$ (6 was already marked).
3. Now the list of unmarked numbers is $5, 7$, so we choose 5 and mark $5, 10$ (10 was already marked).
4. Now the list of unmarked numbers is 7, so we choose 7 and are done.

Implementation hints and improvements:

- Use a `vector<bool>` for marking the numbers up to n.
- Obviously, you do not have to search the next unmarked number from 2 on every time, you can start at $p+1$.
- When marking the multiples of the next prime number p, it is sufficient to start with p^2, i.e., we mark $p \cdot p, (p+1) \cdot p, (p+2) \cdot p, \ldots$.