Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Theoretische Informatik
Dr. B. Gärtner, Prof. Dr. J. Hromkovič

20. November 2007

Informatik für Mathematiker und Physiker Serie 9 HS 07

URL: http://www.ti.inf.ethz.ch/ew/courses/Info1_07/

Skript-Aufgabe 69 (4 Punkte)

What are the problems (if any) with the following functions? Fix them and find appropriate pre- and postconditions.

a)
```
bool is_even (int i)
{
  if (i % 2 == 0) return true;
}
```

b)
```
double inverse (double x)
{
  double result;
  if (x != 0.0)
    result = 1.0 / x;
  return result;
}
```

Skript-Aufgabe 77 (4 Punkte)

Modify the program `sort_array.C` from exercise 60 of the lecture notes in such way that the resulting program `sort_array2.C` defines and calls a function

```
// PRE: [first, last) is a valid range
// POST: the elements *p, p in [first, last) are
//       in ascending order
void sort (int* first, int* last);
```

to perform the sorting of the array into ascending order. You can use the file `sort_array2_template.C` on the course homepage as a template for your modifications. If you do so, you will be able to test your sorting algorithm by input redirection from a file (you will also find some test files on the homepage). You do this by typing `./sort_array2 < inputfile`, where `inputfile` has to be replaced by an actual file of course.

It may be tempting (but not allowed for obvious reasons) to use `std::sort` or similar standard library functions in the body of the function `sort` that is to be defined. It *is* allowed, though, to compare the efficiency of your `sort` function with that of `std::sort` (which has the same pre- and postconditions and can be used after `include<algorithm>`).

For this exercise, it is desirable (but not strictly necessary) to use pointer increment (`++p`) as the only operation on pointers (apart from initialization and assignment, of course). If you succeed in doing so, your sorting function has the potential of working for containers that do not offer random access.

Skript-Aufgabe 78 (8 Punkte)

A *perpetual calendar* can be used to determine the weekday (Monday, ..., Sunday) of any given date. You may for example know that the Berlin wall came down on November 9, 1989, but what was the weekday? (It was a Thursday.) Or what is the weekday of the 1000th anniversary of the Swiss confederation, to be celebrated on August 1, 2291? (Luckily, this big party will be on a Saturday.)

a) The task is to write a program that outputs the weekday (Monday, ..., Sunday) of a given input date.

   Identify a set of subtasks to which you can reduce this task. Such a set is not unique, of course, but all inividual subtasks should be small (so small that they could be realized with few lines of code). It is of course possible for a subtask in your set to reduce to other subtasks. (Without giving away anything, one subtask that you certainly need is to determine whether a given year is a leap year).

b) Write a program `perpetual_calendar.C` that reads a date from the input and outputs the corresponding weekday. The range of dates that the program can process should start no later than January 1, 1900 (Monday). The program should check whether the input is a legal date, and if not, reject it. An example run of the program might look like this.

```
day =? 13
month =? 11
year =? 2007
Tuesday
```

   To structure your program, implement the subtasks from a) as functions, and put the program together from these functions.

Challenge

This week's opportunity for a challenge is exercise 81 from the lecture notes. You can replace 8 of the normal exercise points by that challenge exercise.

Abgabe: Bis 27. November 2007, 15.15 Uhr.