Informatik für Mathematiker und Physiker   HS12

Exercise Sheet 7

Submission deadline:  3.15pm - Tuesday 6th November, 2012
Course URL: [http://www.ti.inf.ethz.ch/ew/Lehre/Info1_12/](http://www.ti.inf.ethz.ch/ew/Lehre/Info1_12/)

Note: Your programs should be well (logically) structured. You should use functions to achieve this. Badly structured programs will be awarded fewer points.

Assignment 1 - (3 points)

Write a program unique.cpp that implements and tests the following function.

```cpp
// PRE: [first, last) is a valid range and describes a sequence
// of elements that are sorted in nondecreasing order
// POST: the return value is true if and only if no element
// occurs twice in the sequence
bool unique (const int* first, const int* last);
```

Assignment 2 - (8 points)

The \emph{n-queens problem} is to place \(n\) queens on an \(n \times n\) chessboard such that no two queens threaten each other. Formally, this means that there is no horizontal, vertical, or diagonal with more than one queen in it. Write a program checkNQueens.cpp that outputs for a given \(n\) and \(n \times n\) board whether the given board is a solution to the \emph{n-queens problem}. The board is given as \(n\) lines of \(n\) characters where "." represents an empty square and "Q" represents a queen. Furthermore, if the given board does not contain \(n\) queens (i.e. fewer or more), the board is not a solution to a n-queens problem.

Here is an example input and the corresponding output:

```
4

...Q
Q...
..Q.
.Q..
```

This is not a solution to n-queens problem.
You can test your program on the following inputs solutions.in, notSolutions.in. Please, note that the input files contain several inputs that are and are not solutions (respectively) to the n-queens problem.

**Assignment 3 - (5 points)**

One of the most common function in applied mathematics is normal distribution, so it is of a great interest to mathematicians and physicists. Consider the [standard normal distribution](#) defined as

\[
\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}
\]

(1)

where \(e\) is [Euler’s number](#). Suppose that we would like to calculate

\[
\int_{a}^{b} \phi(x) \, dx
\]

(2)

for \(a, b \in [-100, 100]\). In general, the definite integral cannot be computed exactly. However, we can approximate its value using [Simpson’s Rule](#)

\[
\int_{a}^{b} f(x) \, dx \approx \frac{b-a}{6} \left[ f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]
\]

(3)

However, the approximation of the definite integral using the Simpson’s rule may be too rough. We can achieve more precise approximation of the definite integral via applying the Simpson’s rule \(n\) times. How do we do this? We know that

\[
\int_{a}^{b} f(x) \, dx = \sum_{i=0}^{n-1} \int_{a_i}^{a_{i+1}} f(x) \, dx
\]

(4)

where \(a_0 = a, a_n = b\) and \(\forall 0 \leq i < n \cdot a_i < a_{i+1}\). Hence, applying the Simpson’s rule to (4) we get

\[
\int_{a}^{b} f(x) \, dx \approx \sum_{i=0}^{n-1} \frac{a_{i+1} - a_i}{6} \left[ f(a_i) + 4f\left(\frac{a_i + a_{i+1}}{2}\right) + f(a_{i+1}) \right]
\]

(5)

After this short mathematical introduction, you are given the following tasks:

(a) Write a program normalDistribution.cpp that reads from the standard input \(a, b \in \mathbb{R}\) and outputs an approximation of (2) using the Simpson’s rule (3).

(b) Extend the program you wrote in the task (a) so that it does not input only \(a, b \in \mathbb{R}\) but also \(n \in \mathbb{N}\) and computes the approximation of (2) using (5) in the way that \(\forall 0 \leq i < n \cdot a_{i+1} - a_i = \frac{b-a}{n}\); i.e. the interval \([a, b]\) is split into \(n\) intervals \([a_i, a_{i+1}]\) of the same size.
Hints:

- For this task (and from now on, if not explicitly told otherwise), you are allowed to use the library `<cmath>`. What might be interesting for you are the following two functions:
  - `exp(x)` returns the base-$e$ exponential function of $x$.
  - `sqrt(x)` returns the square root of $x$.

- Think in the context of your program, how would you have to modify your program if I changed the definition of the normal distribution? And how about, if I did not want to use Simpson’s rule for numerical approximation of a definite integral to some other rule (e.g. Trapezoidal rule)?

- You can test your program via setting $a$ to -100 and $b$ to some value (for $z$ like in the table) comparing the results you’ve got with the standard normal distribution table.

Challenge - (8 points)

Exercise 54 from the lecture notes.