Exercise 3
Numerical Error, Makefile, Graphical output
(Hand out: 04.05.2020, hand in: 14.05.2020)

Review

1. What does the line
   using namespace std ;
   in a C++ program mean? (1 P)
   □ It loads the input/output library.
   □ It allows the usage of global variables
     with names that contain spaces.
   □ It sets the namespace to std, so that
     one can write, e.g., cout instead of
     std::cout.

2. How must variables be passed to functions
   that they can be changed by this function? (1 P)
   □ as a pointer: int *n
   □ as a variable: int n
   □ as a reference: int &n

1. Task Calculating a power series (10 P)
The computation of series for function evaluation is a typical problem that occurs in numerics
and often needs some thoroughly thinking to handle the limitations of finite precision. E.g.,
the exponential
\[ e^{-x} = \sum_{n=0}^{\infty} \frac{(-x)^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots \] (1)

When we want to calculate the exponential exp(−x) for x = 0.1, 1, 10, 100, 1000 we have to stop
the series evaluation at some Nth term. Moreover, the definition of the sum might instruct us
to calculate for the nth term \( x^n \) and \( n! \) and then divide them. This not a good idea. Why? (2P)
Instead, write a program that implements Eq. (1) in a better way to calculate the required
\( x \), i.e. use the fact that for the nth term you’ve already calculated the \( (n - 1) \)th term. Stop
summation for the Nth term when
\[ \left| \frac{(-x)^N/N!}{\sum_{n=0}^{N}(-x)^n/n!} \right| < \epsilon \] (2)

Where \( \epsilon \) is the desired precision.
a) Write a C++ program that produces for a give \( x \) a table of the form
   \[ N \quad \text{sum}(N) \quad \left| \text{sum}(N) - \exp(x) \right| / \exp(x) \]
   where \( \exp(x) \) uses the builtin exponential function. Use \( \epsilon = 10^{-8} \). (3 P)
b) Check that for small \( x \) your algorithm converges, and that it converges to the correct
   answer. (1 P)
c) For increasing \( x \), show that the algorithm converges, but not to the correct value. Show
   that for larger \( x \) not even convergence is given. (1 P)
d) Compare your results (i.e. repeat the steps above) by using the “bad” algorithm. Also,
what happens if \( \epsilon \) is close to the machine precision? How does your algorithm improve
when you avoid subtractive cancellation? (3 P)
2. **Task** *Graphical output in an X window and a simple makefile* (4 P)

For showing animated output of our simulations we will use the X11-based library (it is more like a wrapper) `Xgraphics`. All necessary files may be found here.

a) Compile and run the *first* demo program shown in section 5 of the `Xgraphics` manual. (2 P)

Try to understand the basic elements of this example.

*Heads up!* The given example in the German manual contains a wrong definition of the coordinate system. Instead of

```c
-1,-1,1,1 /* wx1, wy1, wx2, wy2 */
```

it must be:

```c
-1,1,1,-1 /* wx1, wy1, wx2, wy2 */
```

For compilation and linking you can use the following syntax:

```bash
 g++ -c Xgraphics.c
g++ -o program Xgraphics.o program.cpp -lX11
```

b) Create a makefile that executes these commands when `make` is called. Implement also `make clean`.

Important note: For efficient and meaningful use of `make` you should get used to create a separate directory for each project. (2 P)

c) As a preparation for the next programming task:

Write a program that shows an animated point on a circular orbit. (3 extra P)

---

You can also find `Xgraphics.h`, `Xgraphics.c` and the (German) manual `Xgraphics.pdf` in my home directory at `~htodt`. 