Technical University of Munich
TUM School of Engineering and Design
Prof. Dr. Martin Werner

Computational Foundations I (WiSe 2023/2024)
Task Sheet 1

Tasks marked with a star like **Optional Task*** are optional. Tasks marked like **Hard Task+** are given, but it is not expected that you solve them now. It is great if you learn to solve them during the lecture. Go back to them after a few weeks and see your own progress.

Learning Outcome: You will install Visual Studio and desktop C/C++ developing environments and verify a successful installation for following lectures, tutorials and self-study. You will also use Niki the robot to learn imperative programming, i.e., the procedure of telling the computational system what to do in steps.

Task 1: Visual Studio Installation and Verification

Please following the steps given in the slide of the first tutorial `CF1_T1_Xuanshu_231017.pdf` (you can find it in the Moodle page for *Computational Foundations I - Exercise - Tutorial Content*).

Task 2: Niki the robot I

Niki is a traditional environment to teach programming. It is about a small robot that has a very limited set of sensors and actions. However, Niki can be programmed in various ways to solve quite complex problems. As Niki does not have a concept of variables, it uses the environment of the robot to store information.

Task 2.1 Doubling Numbers

Program Niki to initialize a vertical tower of deposited markers starting at location 1/1. The height of the tower shall be interpreted as an natural number. Assure and assume that the robot is located on the basement of the first tower looking right. Now, *write a program* (decompose it into procedures where sensible) to double the number by creating a tower right of the given tower with double the height. (Tip: check the last part of the lecture slide pack 01).

Note that following tasks are taken from the Niki manual. They have been formulated by various people, but not from us.

Task 2.2 Staircase

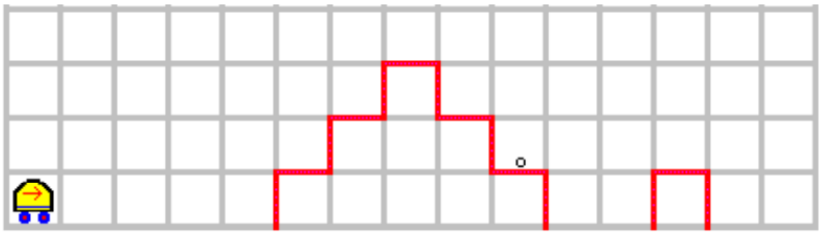

Figure 1: The world for Task 2.2

Recreate the world given in Figure 1 (click on Arbeitsfeld ->Verändern), now the mouse will show you what changes a click would do. These include

- placing markers
- placing walls
- placing the robot

a. *Write a program* that knows the world and picks up the marker, brings it to the podest and deposits it there.

b. Based on domain knowledge that there is first a double staircase and then a podest. *Write a program* that can find the marker on the podest without knowing the detailed shape or position of the marker and bring it to the podest.

Task 2.3 Storage

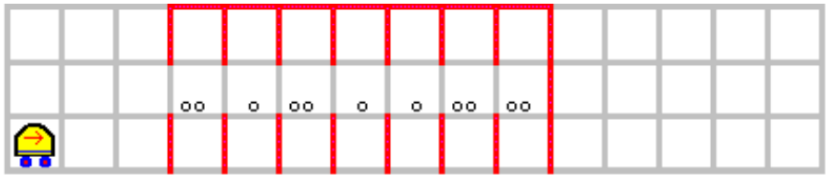

Figure 2: The world for Task 2.3

Niki is supposed to sort the large items (two markers) into the bottom line of the depicted storage space and the small ones (one marker) into the upper row. *Write a program* to implement this function. In a first version, he keeps the X location of all of those. In a more advanced version, the result is supposed to be that the both rows are filled from left to right.

Task 2.4 Waste Collection

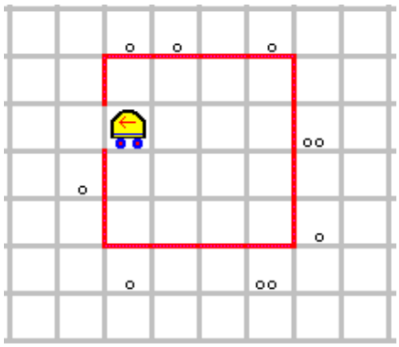

Figure 3: The world for Task 2.4

Niki is somewhere within a hall. The hall has one single entrance. Along the outside wall, there is waste deposited. Niki is supposed to pick up all the waste (not knowing where exactly it is, just that it is adjacent to the wall). *Write a program for it.* (Tip: Try to program it such that left of Niki is always a wall. This is called left hand rule and would walk around any polygons inner or outer boundaries. With the one hole, this rule would change from the inner to the outer boundary and back).This tip enlightens us with a simple fact: sometimes very hard programs can be written pretty concisely by finding an invariant (a natural language condition that remains true throughout the program or program part like a loop). Then, we do a mathematical proof that this invariant is exhausting the problem space (no programming needed) and implement the program (only make sure the invariant is never broken). This approach leads to a notion of algorithmic correctness we will discuss later.