Worksheet 9.1 / Recovering the Flight Recorder

Ceebot Environment:
„Chapter 9: Functions“ / „Flight Recorder“

Mission:
The flight recorder (category `BlackBox`) of a crashed space ship has been located on an island 60 meters east of our base. Write a program for the jet transporter to recover the flight recorder and bring it to the target area.

Concept:
1. Fly 60 meters
2. Grab flight recorder
3. Turn by _____°
4. Fly _____ meters
5. Drop flight recorder

Steps 2, 3 und 5 are pretty simple, each one of them corresponds to a single Ceebot command. In contrast, steps 1 and 4 require more than a single command - takeoff and landing with the _____ - command is a bit tricky, as we already know from Worksheet 8.1. As steps _____ and _____ are identical, it seems to be smart to combine all the actions needed to „Fly 60 meters“ to a new command named `fly` and to invoke this command twice in our program.

Functions combine several commands for a more complex operation.

Working with a function involves two steps:

The **definition of the function** determines the commands that are subsumed under the function’s name. In the program code, the definition can be located before or after the main program.
The **function call** in the main program results in the execution of these commands.

Listing 9.1.1 – Flight Recorder	
Programmcode	**Beschreibung**
`extern void object::FlightRecorder() {`	begin of the main program
`  fly();`	function call of `fly`
`  _____`	
`  _____`	
`  _____`	second function call of `fly`
`  _____`	
`}`	end of main program
`void fly(){`	begin of definition of `fly`
`  jet(1);`	set jet engine to “climb flight”
`  _____`	wait for one second
`  _____`	set jet engine to “hold level”
`  move(_____);`	
`  _____`	set jet engine to “dive”

_____
jet(0);
}
	wait for one second
	_____
	end of definition of `fly`

Two more explanations regarding the header `void fly()`:

- `void` indicates that the function `fly` does not send any information back to the main program.
- The empty parentheses already known from the commands `grab()` und `drop()` indicate that the main program does not pass on any parameters to the function `fly`. Parameters are not necessary as the flight distance always amounts to 60 meters.

Ceebot environment:
„Chapter 9: Functions“ / „Flight Recorder“

Mission:
On the islands towards north, there are three more flight recorders. Recover all four flight recorders and place them onto the landing platform as shown in the picture.

Concept:
As the flight distances no longer amount to 60 meters, we need to redesign the function `fly` for more flexibility: As with the command `move(…)` we want to determine the distance by a parameter. Therefore, we have to modify the definition as well as the function call of `fly`:

- with `void fly(float dist)` we define that the main program has to pass a decimal number (a variable of type _____) as a parameter to the function. **Within the function**, this decimal number is stored in a variable named `dist`. **Outside of the function**, the variable `dist` is not defined.
- As the flight distance is determined by the variable `dist`, the command `move(60)` has to be modified to `move(_____)`.
- As `fly` requires the flight distance as a parameter, the function calls in the main program also have to be altered. If we store the distance to the next flight recorder in a variable named `distToBox`, the part of the main program handling the flight from the landing platform to the next flight recorder might be something like:

```
object item=radar(BlackBox);
turn(direction(item));
float distToBox=distance(this.position,item.position);
fly(distToBox);
```

- If we use the `radar` command exactly as shown above, we will have a problem: `radar` will locate the flight recorder next to the robot, which may be a flight recorder that has just been recovered and placed onto the platform. Therefore you have to alter the `radar`-command in such a way that only objects with a minimum distance of 10 meters are located.

Restart the exercise and reprogram the robot to recover all four flight recorders. In order to place them neatly onto the platform, you might want to use `this.orientation`, which tells you which direction the robot is facing after landing on the platform.

Further exercises:
Write a function `centered`, which gets a string parameter passed from the main program. This string has to be printed in a centered position on the alphanumerical display by using the `printf` command. The alphanumerical display is 80 characters wide; and the length of the string can be determined with `strlen(...)` (see description in Ceebot Help).