Karel Programming Language

Procedure definition:

```
DEFINE proc_name
    commands
END
```

Main procedure declaration (with which procedure should Karel start):

```
RUN proc_name
```

The commands are written on a single line.
Simple commands:

- `SKIP` – Do nothing.
- `STEP` – Move Karel one place forward. If there is a wall, Karel breaks.
- `LEFT` – Turn Karel 90 degrees to the left.
- `RIGHT` – Turn Karel 90 degrees to the right.
- `TAKE` – Have Karel take one beeper from the current place. If there are no beepers, Karel gets confused and breaks.
- `PUT` – Have Karel put one beeper on the current place. (Karel has an unbounded amount of beepers.)
- `proc_name` – Call the given procedure. The call may be recursive.

Branching:

- `IFWALL cmd1 cmd2` – If Karel is facing a wall, do `cmd1`, else do `cmd2`.
- `IFMARK příkaz1 příkaz2` – If there is at least one beeper on the current place, do `cmd1`, else do `cmd2`.

In both cases, `cmd1` and `cmd2` need to be simple commands, not another branching commands.
Comments start with `#` and end with the end of line.

City Specification

The first line gives the size of the city: the number of rows followed by the number of columens. The second line gives the position of Karel (in the form `y x`) follow by Karel's orientation (`n`, `e`, `s`, nebo `w`). The following lines describe the city. Every place is either a number (a non-wall place with the given amount of beepers) or the `#` character meaning a wall. The city is implicitly surrounded by walls in all directions.
City specification example:

```
7 7
3 3 n
0 0 0 0 0 0 0
0 # # 0 # # 0
0 # # 0 # # 0
0 0 0 0 # 2 #
0 0 0 # # 0 #
# # 0 # 0 0 #
0 0 0 0 0 # 0
```

Running the interpreter (needs Python $\geq$ 3.5)

```
$ python karel.py city program [-p]
```

`city` is a city specification file, `program` is a program source file, `-p` is an optional flag that causes the interpreter to stop and wait for *Enter* after each step.