ANSWERS

2.4 Getting round things

Sensing · Robot club · about 20 min

What this lesson is about

Detect, sidestep, continue: the first obstacle avoidance.

- The plan
- Step 1: approach
- Step 2: sidestep until clear
- Step 3: past and on
- Which way to step?

Questions

6 marks in all

1. Put the steps for getting round a box in order. [1 mark]

Number the lines 1 to 5 to put them in the right order.

___ `Stop`
___ `Slide right until distance() is over 60 cm`
___ `Drive forward 35 cm to get past`
___ `Slide right 6 cm more`
___ `Drive forward until distance() is under 20 cm`

Answer:

```
Drive forward until distance() is under 20 cm
Stop
Slide right until distance() is over 60 cm
Slide right 6 cm more
Drive forward 35 cm to get past
```

Approach, step sideways until the way is clear, a bit extra for the body, then drive past. The sensor decides when each step is done.

2. Why does the program slide another 6 cm after the sensor says the way is clear? [1 mark]

- ☐ **A** The body is wider than the sensor's beam
- ☐ **B** The sensor is always 6 cm wrong
- ☐ **C** To make up for coasting forward
- ☐ **D** To turn the robot back to heading 0

Answer: A. The sensor looks out from the middle, so it can see past the box before the edge of the robot is clear of it.

3. Why does sliding sideways, rather than turning, keep this program simple? [1 mark]

- ☐ **A** The robot never turns, so "ahead" still means the same direction
- ☐ **B** Sliding is faster than turning
- ☐ **C** The sensor does not work after a turn
- ☐ **D** Turning would reset position()

> **Answer:** A. After strafing, `distance()` still looks the way the robot is going, so the same loops work before and after.

4. What does this loop do? [1 mark]

```
while distance() < 60:
    right(50)
    wait(0.1)
stop()
```

- ☐ **A** Slides right for as long as something is closer than 60 cm ahead
- ☐ **B** Slides right exactly 60 cm
- ☐ **C** Slides right until something is closer than 60 cm
- ☐ **D** Drives forward for 60 cm

> **Answer:** A. `while distance() < 60` keeps going while the way ahead is blocked. It stops once the sensor sees further than that.

5. What does this program print? [1 mark]

```
left_side = 25
right_side = 25
if right_side >= left_side:
    print("stepping right")
else:
    print("stepping left")
```

> **Answer:**
>
> ```
> stepping right
> ```
>
> 25 is equal to 25, and `>=` is true for equal numbers, so a tie goes right.

6. What could go wrong if the robot always steps right? [1 mark]

- ☐ **A** There might be no room on the right
- ☐ **B** It would turn round
- ☐ **C** The distance sensor would stop working
- ☐ **D** Nothing, stepping right always works

> **Answer:** A. The depth grid's edge columns tell you which side has more room, so the robot can pick the better way.

The task: dodge the box

Drive until the box is close, print `box ahead`, step sideways until the way is clear, go past, and finish in the green zone without touching the box.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

# as long as the thing ahead is further than 20 cm
while distance() > 20:
    # drive forward at 50 (keeps going until the next command)
    forward(50)
    # pause 0.1 s (the robot keeps doing what it was told)
    wait(0.1)
# all motors off
stop()
print("box ahead")
```

The hint students can ask for: Drive until distance() says the box is close, print `box ahead`, step sideways until the way is clear, go past, and into the green zone.

A solution

```
from bugbot import *
connect()
while distance() > 20:
    forward(50)
    wait(0.1)
stop()
print('box ahead')
while distance() < 60:
    right(50)
    wait(0.1)
stop()
right(50, distance=6)                # a little further: the body is wider than the
sensor's beam
forward(60, distance=55)
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.