

Exercise 5: Parsing Directions

Assume we have:

- `next_token();`
- `advance();`
- `bool next_token_is_number();`
- `fail("message");`
- `expect(target);` // if the next token is the target, then advance, otherwise fail

Directions → 'go' SegmentDirection CantMiss

An instruction specifying that you should follow certain directions with a notification of not being able to miss something.

```
void parse_directions() {
    expect("go");
    parse_segment_direction();
    parse_cant_miss();
    if (next_token() != null) {
        fail("extra tokens at end of parse directions");
    }
}
```

SegmentDirection → Segment {',' Segment }* ','

A comma separated list of segments that should be moved that contains at least one segment.

```
void parse_segment_direction() {
    parse_segment();
    while (next_token() == ",") {
        advance();
        parse_segment();
    }
    expect(".");
}
```

Segment → 'left' | 'right' | 'straight' Number 'blocks' | StopAt

A movement instruction to turn left, turn right, go straight several blocks or stop somewhere.

```
void parse_segment() {  
    string curr_token = next_token();  
    if (curr_token == "left" || curr_token == "right") {  
        Advance();  
    } else if (curr_token == "straight") {  
        advance();  
        parse_number();  
        expect("blocks");  
    } else {  
        parse_stop_at();  
    }  
}
```

Number -> [0-9]+

One or more digits 0-9.

```
void parse_number() {  
    if (!next_token_is_number()) {  
        fail("expected a number");  
    }  
    while (next_token_is_number()) {  
        advance();  
    }  
}
```

StopAt -> 'stop' 'at' ('sign' | 'light' | 'destination')

Stop instruction of the form: "stop at " followed by one of the following: 'sign', 'light', or 'destination'.

```
void parse_stop_at() {
    expect("stop");
    expect("at");

    if (next_token() == "sign" || next_token() == "light" || next_token() ==
        "destination") {
        advance();
    }
    else {
        fail("expected one of the following: 'sign','light','destination'");
    }
}
```

CantMiss -> 'you' 'cannot' 'miss' 'it' '.' | 'and' 'you' 'are' 'there' '.'

Terminal sequence that represents the end of the directions

```
void parse_cant_miss() {
    current_token = next_token();
    if (current_token == "you") {
        advance();
        expect("cannot");
        expect("miss");
        expect("it");
        expect(".");
    }
    else if (current_token == "and") {
        advance();
        expect("you");
        expect("are");
        expect("there");
        expect(".");
    }
    else {
        fail("expected : 'you' or 'and' ");
    }
}
```