Multiplication quiz: a game to practice "while" loop

David Zelený & Po-Ju Ke

Aim: Use the `while` loop to simulate the following *multiplication quiz* game, in which the user (you or anybody using the computer) is quizzed about her ability to multiply two numbers by her head. The game goes like this:

1. The computer randomly generates two numbers, each being an integer between 1 and 9. Let's say that in the first step, it generates numbers `8` and `9`. The user is being asked: `How much is 8*9?`
2. The user types the answer which s/he considers correct (e.g. `56`) into the command line and presses enter.
3. The computer evaluates whether the answer is correct; if yes, it will reply, `Congratulations, correct answer!` and stops the game. If not correct, it will reply, `Sorry, the wrong answer, the correct answer is 72. You have another try.` After this, the game runs from the beginning (step 1).

Please wrap the whole game into a function called `multiplication.quiz`. The function should have no additional arguments. It should run as the followings:

```
> multiplication.quiz ()
How much is 8*9? 56
[1] "Sorry, wrong answer, correct is 72. You have another try."
How much is 4*9? 69
[1] "Sorry, wrong answer, correct is 36. You have another try."
How much is 7*2? 13.5
[1] "Sorry, wrong answer, correct is 14. You have another try."
How much is 3*4? 12
[1] "Congratulations, correct answer!"
```

Hints

1. You will need the function `readline`, which can read the response as it was typed into the command line (console); example: `year <- readline ('Which year you got born?')` should prompt you with this question, and you answer typed after the question will be stored into the `year` variable (note that even if the user types in the number (e.g. 2001), the value gets stored as a character string (ie "2001") and if you want to use it for calculation, you need to convert it into a number (function `as.numeric`).
2. You will also most probably need `if` and `else` statements (or something similar).
3. The `paste` function can assemble the sentence which contains several components (e.g. `x <- 2; paste ('This is', x, sep = ' ')` returns `This is 2`).