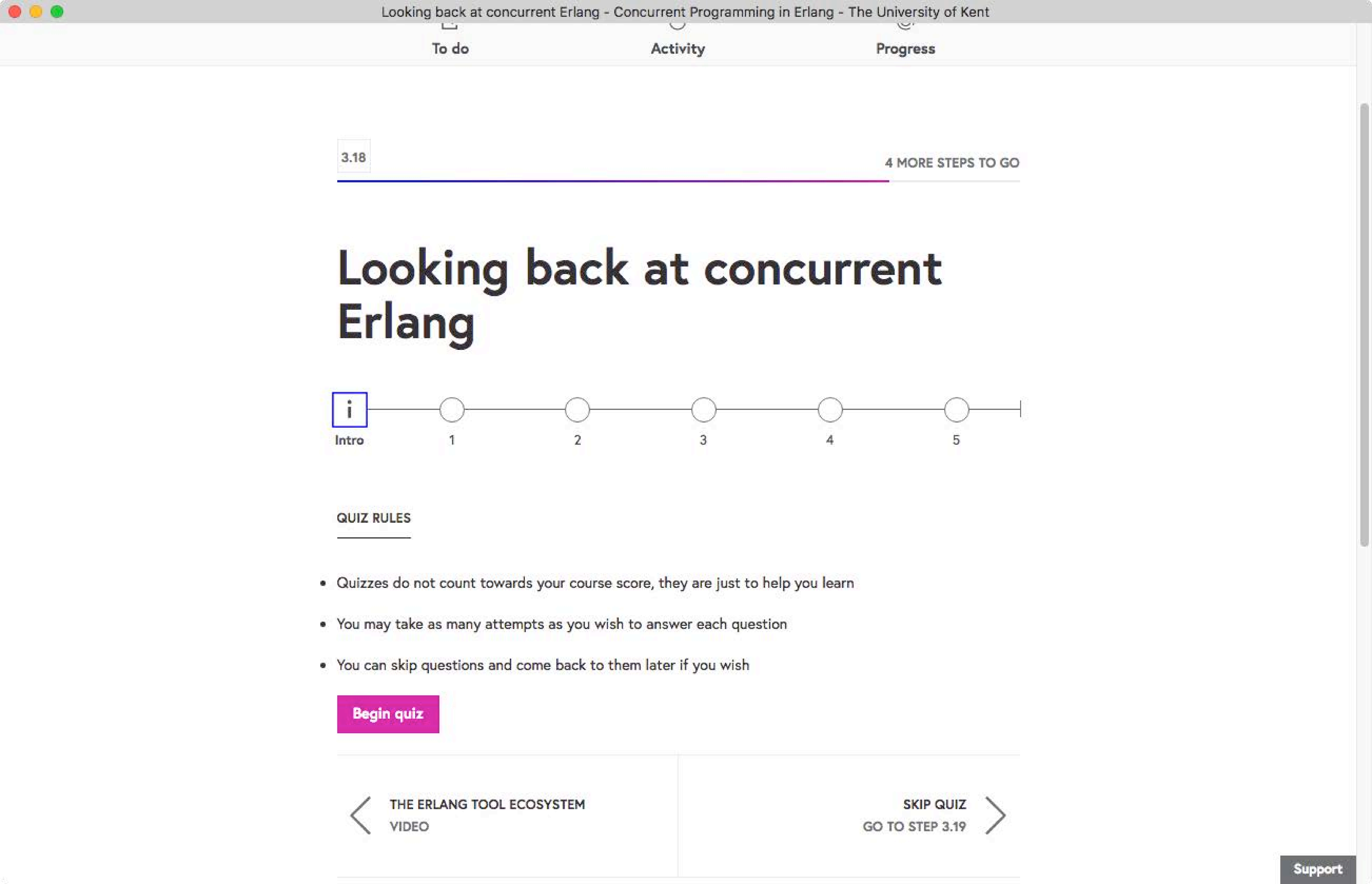


3.18 4 MORE STEPS TO GO

Looking back at concurrent Erlang

QUIZ RULES

- Quizzes do not count towards your course score, they are just to help you learn
- You may take as many attempts as you wish to answer each question
- You can skip questions and come back to them later if you wish

THE ERLANG TOOL ECOSYSTEM
VIDEO

SKIP QUIZ
GO TO STEP 3.19

Looking back at concurrent Erlang

Intro 1 2 3 4 5

Question 1

When receiving a message via the `receive` construct, what happens if none of the clauses matches the incoming message?

- ○ The receiver process will crash due to a `case_clause` error
- ○ The very last clause will be matched (the last clause is always a "catch all" clause)
- ○ The message will be kept in the mailbox of the process and any messages received subsequently will be pattern-matched against the clauses.
- ○ The message will be removed from the mailbox.

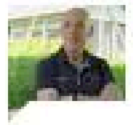
Support

Question 1

When receiving a message via the `receive` construct, what happens if none of the clauses matches the incoming message?

- The receiver process will crash due to a `case_clause` error
- The very last clause will be matched (the last clause is always a "catch all" clause)
- [✓ **The message will be kept in the mailbox of the process and any messages received subsequently will be pattern-matched against the clauses.**]
- The message will be removed from the mailbox.

Correct



Simon Thompson LEAD EDUCATOR

Yes, that's correct.

Support

Question 2

What is the effect of typing

```
receive X -> X end.
```

to the `erl` prompt, assuming that the following interaction as already taken place:

```
1> self() ! hello.
hello
2> receive X -> X end.
hello
3> self() ! goodbye.
goodbye
```

- ○ The program prints `hello`.
- ○ The program hangs.
- ○ The program prints goodbye.
- ○ The program crashes.

Support

Question 2

What is the effect of typing

```
receive X -> X end.
```

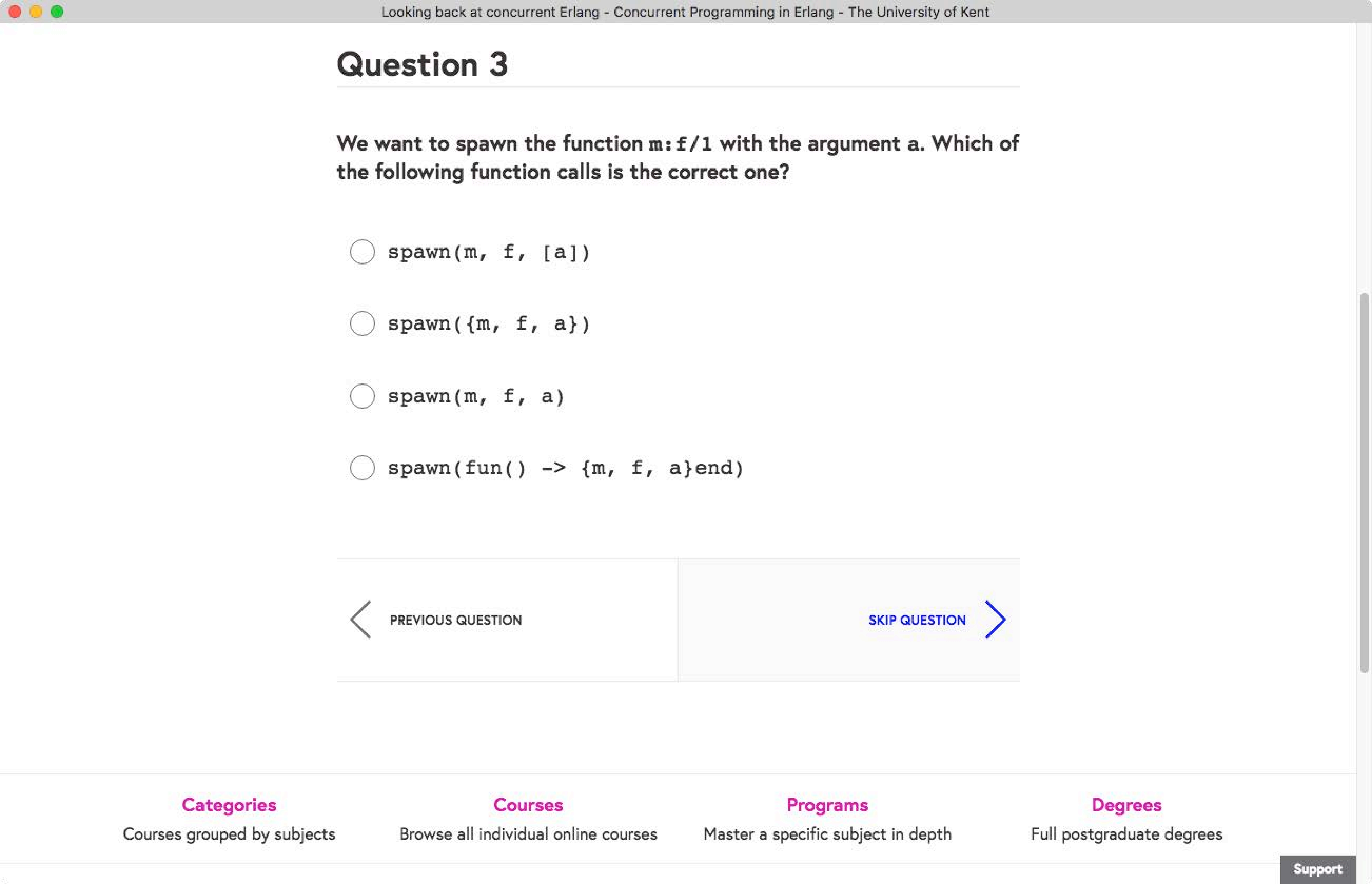
to the `erl` prompt, assuming that the following interaction as already taken place:

```
1> self() ! hello.
hello
2> receive X -> X end.
hello
3> self() ! goodbye.
goodbye
```

- ○ The program prints `hello`.
- [✓ **The program hangs.**]
- ○ The program prints `goodbye`.
- ○ The program crashes.

Correct

Support



Question 3

We want to spawn the function `m:f/1` with the argument a. Which of the following function calls is the correct one?

- ◯ `spawn(m, f, [a])`
- ◯ `spawn({m, f, a})`
- ◯ `spawn(m, f, a)`
- ◯ `spawn(fun() -> {m, f, a}end)`

Question 3

We want to spawn the function `m:f/1` with the argument a. Which of the following function calls is the correct one?

- [✓ `spawn(m, f, [a])`]
- `spawn({m, f, a})`
- `spawn(m, f, a)`
- `spawn(fun() -> {m, f, a}end)`

Correct

Simon Thompson LEAD EDUCATOR

Yes, that's correct.

PREVIOUS QUESTION

NEXT QUESTION

Support

Question 4

Which of the following statements concerning *trapping exits* are false?

- [ ] When a process P is not trapping exits and a process Q linked to it terminates, it will terminate too, independently of the reason for termination of Q.
- [ ] A process can be set to trap exit signals by calling `process_flag(trap_exit, true)`.
- [ ] When a process is trapping exits, it will terminate when an exit signal is received.

PREVIOUS QUESTION

SKIP QUESTION

Question 4

Which of the following statements concerning *trapping exits* are false?

- [✓] **When a process P is not trapping exits and a process Q linked to it terminates, it will terminate too, independently of the reason for termination of Q.**
- [ ] A process can be set to trap exit signals by calling `process_flag(trap_exit, true)`.
- [ ] When a process is trapping exits, it will terminate when an exit signal is received.

Correct

Simon Thompson LEAD EDUCATOR

Yes, this is false because it in the case of normal termination of Q, P will not terminate.

Support

Question 5

Code is loaded in the run time system by:

- ☐ Explicitly loading it using `code:load_file(Module)`.
- ☐ Calling a function in a module which is not already compiled.
- ☐ Calling a function in a module which is not already loaded.
- ☐ Calling the shell function c to compile the module.

PREVIOUS QUESTION

FRANCESCO, JOE AND SIMON: ERLANG/OTP NOW AND IN THE FUTURE
VIDEO

Question 5

Code is loaded in the run time system by:

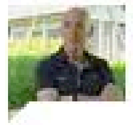
[✓ Explicitly loading it using `code:load_file(Module)`.]

Calling a function in a module which is not already compiled.

[✓ Calling a function in a module which is not already loaded.]

[✓ Calling the shell function c to compile the module.]

Correct



Simon Thompson LEAD EDUCATOR

Yes, that's correct.

Yes, this has the result of loading a module.

Yes, calling the shell compile function has the effect of loading the module.

Support