

## Solutions for function

### functions

1.

See source code file invoke1.c

The compiler did produce a program for us. The compiler did however warn us.

2.

See source code file invoke1.c

The compiler did produce a program for us. The compiler did however warn us, just as in (1).

3.

See source code file invoke1.c

The compiler did NOT produce a program for us. The compilation failed.

4.

See source code file invoke1.c

The compiler did produce a program for us. The compiler did NOT warn us.

5.

See source code file invoke5.c

6.

See source code file invoke6.c

7.

I refuse to write this code.

8.

See source code file invoke8.c

9.

See source code file invoke-return9.c

10.

The function returns an int. The return type of the function is placed before the name of the function and since the function (so called) prototype looks like this:

```
int gimme_three(void)
```

we can see that the function returns an int.

11.

See source code file invoke-return11.c

A good type to store the returned value is an int, since the returned value is of the type int.

12.

See source code file invoke-return12.c

13.

The method takes two integer parameters (arguments). The functions internal name for the parameters are **a** and **b**. These parameters are added and returned. So, in short, the function returns the sum of two parameters.

14.

See source code file invoke-return15.c

15.

See source code file invoke-return15.c

16.

Let's look at the code for a while:

```
result = add(add(12, 34), add(34,56));
```

In order to invoke the outermost add we need to know what parameters to pass to the function. In order to know this we need to evaluate the expressions `add(12, 34)` and `add(34,56)`. These expressions are evaluated to: 46 and 90, so the outermost call to add really is `add(46,90)`. If we evaluate that we get 136.

17.

Answer is in the question.

18.

Answer is in the question.