```
/*
This shows you how to use other cogs.
What you need to do is
1. Create a function
2. Send the function to another cog to run but store a pointer to the cog.
3.Stop the other cog  (using the pointer you stored in step 2).
(Stopping the other cog will also release
the memory space it is using.)
*/

#include "simpletools.h" // Include simple tools

// here is the function to be run in another cog

blink()
{
while(1)
{
high(27);
pause(500);
low(27);
pause(500);
} //end while
} // end blink



int main()
{
/* send the function to a new cog
store the cog info in a variable so it can be stopped
&blink is the address of the function, 10 is how much stack space to
allocate.
blinker is a pointer variable to use when stopping the cog.
*/
int *blinker =cog_run(&blink,10);
pause(8000); //run for a while or do other stuff here
cog_end(blinker); // then stop the cog and release the memory
}
```

Rules for cog_run:

The cog_run function itself needs two parameters:

- **&function**, which is the address of the function you want to launch (&blink in this example)
- **stackSize**, a value to set aside additional memory called *stack space* for the cog to perform its computations (10 in this example means a stack space of 10 32-bit memory locations. It is used for performing calculations while executing the instructions in the blink code block.).

A function launched with cog_run:

- Can not require parameters passed to it (use global variables, if needed)
- Can not return a value (again use global variables if needed)
- Should not contain a print, scan, or other function call that uses the SimpleIDE Terminal, unless your program is going to specifically manage that with additional functions.

Stack Size - how much? 10 is the bare minimum value you would want to use for the **stackSize** parameter. If you were to add more instructions to the blink function's code block, you would need to increase it. Add 1 for every local variable used, 2 for each function called, and 1 for each parameter and each return value used by the functions called.