"Hello world" and error checking with `compute-sanitizer`

Exercise 2:

Write a simple OpenMP offload "hello world" code which produces output like

```
...
Hello world! I'm thread 30 out of 64 in team 0. My global thread id is 30 out of 256.
Hello world! I'm thread 31 out of 64 in team 0. My global thread id is 31 out of 256.
Hello world! I'm thread 0 out of 64 in team 2. My global thread id is 128 out of 256.
...
```

The output is deterministic for groups of 32 lines, but otherwise non-deterministic.

A Makefile template is provided on DTU Learn.

1. Experiment with the number of threads per team and number of teams. Is the output deterministic? Run with `./helloworld | wc -l` to count the number of lines.

Note: If you offload code that gives errors (like segmentation faults etc.) on the target device, **you will likely not receive useful error messages** to help you debug.

The remainder of this exercise illustrates a simple way to obtain better error information when offloading.

2. Modify your OpenMP offload code so that the thread with global thread id `tid = 100` makes a segmentation fault just before it prints out the "Hello world!". Simply try to access memory you haven't allocated, e.g., with the following buggy code:

   ```
   if (global_tid == 100) { int *a = (int*) 0x10000; *a = 0; }
   ```

   What change do you see in the output - how many hellos are missing? Why?

3. Note that the segmentation fault gives a simple NO_SUCCESS (value 1) error. Try to catch the segmentation fault and get more information by using

   ```
   compute-sanitizer ./helloworld
   ```

   when you run your program. For more information about the output and usage, see `http://docs.nvidia.com/compute-sanitizer/ComputeSanitizer/index.html`