```
Small demos of model checking.

Model check C source code using cbmc
---------------------------------------------------------------

// test0.c
int main()
{
  int i;
  int s = 0;
  for (i=0; i&lt;10; i++) s += i;
   __CPROVER_assert(s > 200, "my postcondition");
   return 0;
}


$ cbmc test0.c
...

** Results:
[main.assertion.1] my postcondition: FAILURE

** 1 of 1 failed (1 iteration)
VERIFICATION FAILED

---------------------------------------------------------------
Model check over undefined inputs:


extern unsigned int reader();

int main()
{

  int i, N = 2 + (reader()&0xFFFffff);
  int ss = 0;
  for (i=0; i<N; i++) { ss += 3*i; }
   __CPROVER_assert(ss >=3, "postcondition");
   return 0;
}


$ cbmc test1.c --unwind 1000

Other model checkers do not require an unwind limit.  But the cbmc
checker is a bounded model checker and looks only for
counterexamples up to a given number of applications of the
next state operator from operational semantics.

---------------------------------------------------------------
// test2.c.
// Here we see the basic C construct for thread spawning.
// cbmc models this and can consider all possible interleavings of the threads.
#include <pthread.h>
#include <stdio.h>
#include <assert.h>

int sv; // A shared variable

void *producer()
  {
    while (sv &lt; 100) sv += 3;
  }
```

```
void *consumer()
  {
    while (sv &lt; 100) sv += 5;
  }


pthread_t tid0, tid1;

int main()
{
  pthread_create(&tid0, NULL, producer, (void *)0);
  pthread_create(&tid1, NULL, consumer, (void *)0);
  pthread_join(tid0, 0);
  pthread_join(tid1, 0);
  printf("sv is %i\n", sv);
  assert(sv==100);
  return 0;
}


---------------------------------------------------------------

... under construction

---------------------------------------------------------------

... under construction
```