```
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_spawn$ ./a.out
thread #0 id = 0x1041f0000
thread #1 id = 0x104273000
thread #2 id = 0x1042f6000
thread #3 id = 0x1041f0000
thread #4 id = 0x104273000
thread #5 id = 0x1042f6000
thread #6 id = 0x1041f0000
thread #7 id = 0x104273000
thread #8 id = 0x1042f6000
thread #9 id = 0x1041f0000
Actual thread spawned = 3
0x1041f0000
0x104273000
0x1042f6000
```

Possible output

```
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_spawn$ ./a.out
thread #0 id = 0x1041f0000
thread #1 id = 0x104273000
thread #2 id = 0x1042f6000
thread #3 id = 0x1041f0000
thread #4 id = 0x104273000
thread #5 id = 0x1042f6000
thread #6 id = 0x1041f0000
thread #7 id = 0x104273000
thread #8 id = 0x1042f6000
thread #9 id = 0x1041f0000
Actual thread spawned = 3
0x1041f0000
0x104273000
0x1042f6000
```

std::async

```
template< class Function, class... Args >
std::future<typename std::result_of<Function(Args...)>::type>
async( std::launch policy, Function&& f, Args&&... args );
```

- The template function async runs the function f asynchronously (potentially in a separate thread which may be part of a thread pool) and returns a std::future that will eventually hold the result of that function call.
- Policies to spawn computation are:
  - std::launch::async
  - std::launch::deferred
  - launch::any (bitwise or async | deferred)

from http://en.cppreference.com/w/cpp