- Exercise

In this exercise, you will throw and handle an exception using std::promise and std::future.

```
we'll cover the following ^
• Task
```

Task

Implement a program where the promise throws an exception and that exception is handled in the associated future. As a starting point, you can use the code from the previous lesson and invoke it with the denominator.

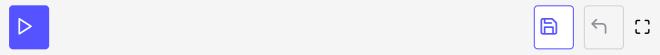
```
// promiseFuture.cpp
                                                                                           G
#include <future>
#include <iostream>
#include <thread>
#include <utility>
void product(std::promise<int>&& intPromise, int a, int b){
  intPromise.set_value(a*b);
struct Div{
  void operator() (std::promise<int>&& intPromise, int a, int b) const {
    intPromise.set_value(a/b);
};
int main(){
  int a= 20;
  int b= 10;
  std::cout << std::endl;</pre>
  // define the promises
  std::promise<int> prodPromise;
  std::promise<int> divPromise;
  // get the futures
```

```
std::future<int> prodResult= prodPromise.get_future();
std::future<int> divResult= divPromise.get_future();

// calculate the result in a separat thread
std::thread prodThread(product, std::move(prodPromise), a, b);
Div div;
std::thread divThread(div, std::move(divPromise), a, b);

// get the result
std::cout << "20*10= " << prodResult.get() << std::endl;
std::cout << "20/10= " << divResult.get() << std::endl;

prodThread.join();
divThread.join();
std::cout << std::endl;
}</pre>
```



The solution to this exercise is available in the next lesson.