A Counter Monad: Example of Use

```haskell
-- the result is 12.
main myProc:

run the program:

return (c1*c2)
  c2 -> 
  inc c1 <- 
  c1 <- 
myProc :: Counter Int

An example "program" using a counter:

A Counter Monad

The "run" system for our counter monad may look like this:

```
Counter Monad with Exceptions

The runtime system may look like this:

```cpp
void error = throw; exception

// A program that handles an exception:
error = func() {
    try {
        throw exception;
    } catch (exception e) {
        // Handle exception
    }
}
```

To allow the user to catch and handle exceptions:

```cpp
throw exception e; exception

int main() {
    try {
        throw exception;
    } catch (...) {
        // Handle exception
    }
}
```

Where `throw` is defined as a command that throws an exception:

```cpp
throw exception e; exception
```

Counter Monad with Exceptions

Counter Monad with Exceptions

```cpp
Counter Monad with Exceptions

The possible operations:

```cpp
(\text{Counter Monad}) = \text{Counter} \circ \text{Exception}
```

Let's say that will overflow if the counter exceeds 3:

```cpp
Counter Monad with Exceptions

```