5) Exception Handling

```
#include <iostream>
#include <stdexcept>
using namespace std;

template <typename T>
T safeDivide(T numerator, T denominator)
{
   if (denominator == 0)
   {
      throw runtime_error("Division by zero Exception: Attempted to divide by zero!");
   }
   return numerator / denominator;
}

int main()
{
   try
   {
      int result_int = safeDivide(10, 2);
      cout << "Integer Division (10/2): " << result_int << endl;

      result_int = safeDivide(10, 0);
      cout << "Integer Division (10/0): " << result_int << endl;
   }
   catch (const runtime_error &e)
   {
      cout << "Error Caught: " << e.what() << endl;
   }
   cout << endl;
{
   try
   {
      double result_double = safeDivide(15.0, 4.0);
      cout << "Double Division (15.0/4.0): " << result_double << endl;

      result_double = safeDivide(20.0, 0.0);
      cout << "Double Division (20.0/0.0): " << result_double << endl;
   }
   catch (const runtime_error &e)
   {
      cout << "Error Caught: " << e.what() << endl;
   }
   cout << endl;
   try
```

```
    {
        double mixed_result = safeDivide(10, 3.0);
        cout << "Mixed type Division (10/3.0): " << mixed_result << endl;
    }
    catch (const runtime_error &e)
    {
        cout << "Error Caught: " << e.what() << endl;
    }

    return 0;
}
```