1. **Write a c++ program to demonstrate to use of the finally block for handling exceptions**

```
#include <iostream>

class Finally {
public:
    ~Finally() {
        std::cout << "Executing finally block" << std::endl;
    }
};

void functionWithFinally() {
    Finally f;
    try {
        throw std::runtime_error("An error occurred");
    } catch (const std::exception &e) {
        std::cout << "Caught exception: " << e.what() << std::endl;
    }
}

int main() {
    functionWithFinally();
    return 0;
}
```

Output : Caught exception: An error occurred
Executing finally block

2. **Write a c++ program to demonstrate to use of nested try-catch blocks for handling exceptions**

```
#include <iostream>
#include <stdexcept>

void nestedTryCatch() {
    try {
        try {
            throw std::runtime_error("Inner exception");
        } catch (const std::runtime_error &e) {
            std::cout << "Caught inner exception: " << e.what() << std::endl;
            throw;  // rethrow the exception
        }
    } catch (const std::exception &e) {
        std::cout << "Caught outer exception: " << e.what() << std::endl;
```

```
    }
}

int main() {
    nestedTryCatch();
    return 0;
}
```

Output : Caught inner exception: Inner exception
Caught outer exception: Inner exception

3. **Write a c++ program to demonstrate to use of user-defined exception for handling custom exception**

```
#include <iostream>
#include <exception>

class MyException : public std::exception {
public:
    const char* what() const noexcept override {
        return "My custom exception";
    }
};

void functionWithCustomException() {
    throw MyException();
}

int main() {
    try {
        functionWithCustomException();
    } catch (const MyException &e) {
        std::cout << "Caught custom exception: " << e.what() << std::endl;
    }
    return 0;
}
```

Output : Caught custom exception: My custom exception

4. **Write a c++ program to demonstrate to use of the standard class for handling exceptions**

```
#include <iostream>
#include <stdexcept>

void functionWithStandardException() {
```

```
    throw std::runtime_error("Standard runtime error");
}

int main() {
    try {
        functionWithStandardException();
    } catch (const std::runtime_error &e) {
        std::cout << "Caught standard exception: " << e.what() << std::endl;
    }
    return 0;
}
```

Output : Caught standard exception: Standard runtime error

5. **Write a c++ program to demonstrate to use of the keyword to throw an exception**

```
#include <iostream>
#include <stdexcept>

void functionThrowingException() {
    throw std::runtime_error("Thrown exception");
}

int main() {
    try {
        functionThrowingException();
    } catch (const std::exception &e) {
        std::cout << "Caught exception: " << e.what() << std::endl;
    }
    return 0;
}
```

Output : Caught exception: Thrown exception

6. **Write a c++ program to demonstrate to use of multiple catch blocks for handling different** types of exceptions

```
#include <iostream>
#include <stdexcept>

void functionWithMultipleExceptions(int errorType) {
    if (errorType == 1) {
        throw std::runtime_error("Runtime error");
    } else if (errorType == 2) {
        throw std::logic_error("Logic error");
    } else {
```

```
        throw std::exception();
    }
}

int main() {
    try {
        functionWithMultipleExceptions(1);
    } catch (const std::runtime_error &e) {
        std::cout << "Caught runtime error: " << e.what() << std::endl;
    } catch (const std::logic_error &e) {
        std::cout << "Caught logic error: " << e.what() << std::endl;
    } catch (const std::exception &e) {
        std::cout << "Caught general exception" << std::endl;
    }
    return 0;
}
```

Output : Caught runtime error: Runtime error

7. **Write a c++ program to demonstrate to use of try-catch blocks for handling exceptions**

```
#include <iostream>
#include <stdexcept>

void functionWithTryCatch() {
    try {
        throw std::runtime_error("Exception in function");
    } catch (const std::exception &e) {
        std::cout << "Caught exception: " << e.what() << std::endl;
    }
}

int main() {
    functionWithTryCatch();
    return 0;
}
```

Output : Caught exception: Exception in function