# Chapter 10 Programming Assignment Documentation

Ashton Hellwig

May 10, 2020

## Contents



## List of Algorithms



## Listings

# 1 Problem Analysis

The problem states:

> This assignment relates to content from Chapter 10 of the eText.
>
> **Instructions**
>
> 1. Review the general programming assignment instructions.
> 2. Write a program that:
>
>    A. Placeholder.

## 1.1 Data

- Placeholder.

## 1.2 Desired Output

```
1 echo "TODO!"
```

Listing 1: ashellwig_m5c10_programming_assignment output (stdout)

# 2 Algorithm

Below is the algorithm for the program.

**Algorithm 2.1** Chapter 10 Program Algorithm

```
1: procedure MAIN
2:     return 0
3: end procedure
```

# 3 User Documentation

Please see Appendix A for images showing the compilation and running of the program.

## 3.1 Build

The following are instructions with two use cases:

- With GNU Make
- Bundled Release

### 3.1.1 With GNU Make

1. Navigate to the unzipped folder containing the project, **with a terminal emulator or command prompt**, this will (most likely) mean running:

```
cd ~/Downloads/ashellwig_m5c10_programming_assignment
```

2. Compile the program and documentation[1] using GNU automake after switching to the source directory:

```
# precondition: Must be located in project root
export INPUT_FILE="${PWD}/data/bookData.txt"
export OUTPUT_FILE="${PWD}/out/bin/output_data.txt"

make debug

./out/bin/ashellwig_m5c10_programming_assignment.bin \
  -f "${INPUT_FILE}" \
  -o "${OUTPUT_FILE}"


cat "${OUTPUT_FILE}" # Verify contents are present

make clean-all       # Removes object files, binaries, and docs
```

Listing 2: Chapter 10 Program Build Commands

### 3.1.2 Bundled Release

1. Navigate to the unzipped folder containing the binary, **with a terminal emulator or command prompt**, this will (most likely) mean running:

```
cd ~/Downloads/ashellwig_m5c10_programming_assignment
```

2. To run the program simply issue this within the command prompt

```
export INPUT_FILE="${PWD}/data/Ch9Ex2/Ch9Ex2Data.txt"
export OUTPUT_FILE="${PWD}/data/Ch9Ex2OUT.txt"

./out/bin/ashellwig_m5c10_programming_assignment.bin \
  -f "${INPUT_FILE}" \
  -d "${OUTPUT_FILE}"
cat "${OUTPUT_FILE}"  # Verify contents are present
```

[1] **Note**: This requires the whole `texlive` suite as well as `latexmk` to be installed.

Of course if preferred, you may also navigate to the build folder in file explorer and double click the executable (`./ashellwig_m5c10_programming_assignment`).

## Works Consulted

Malik, D. S. (2015). *C programming: Program design including data structures* (7th ed.). Cengage Learning.

# A Images

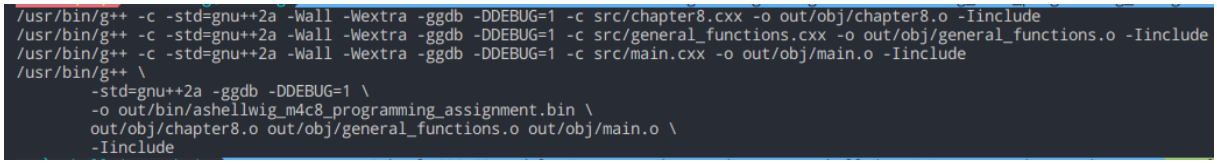



Figure 1: Compiling Chapter 10‘s Program

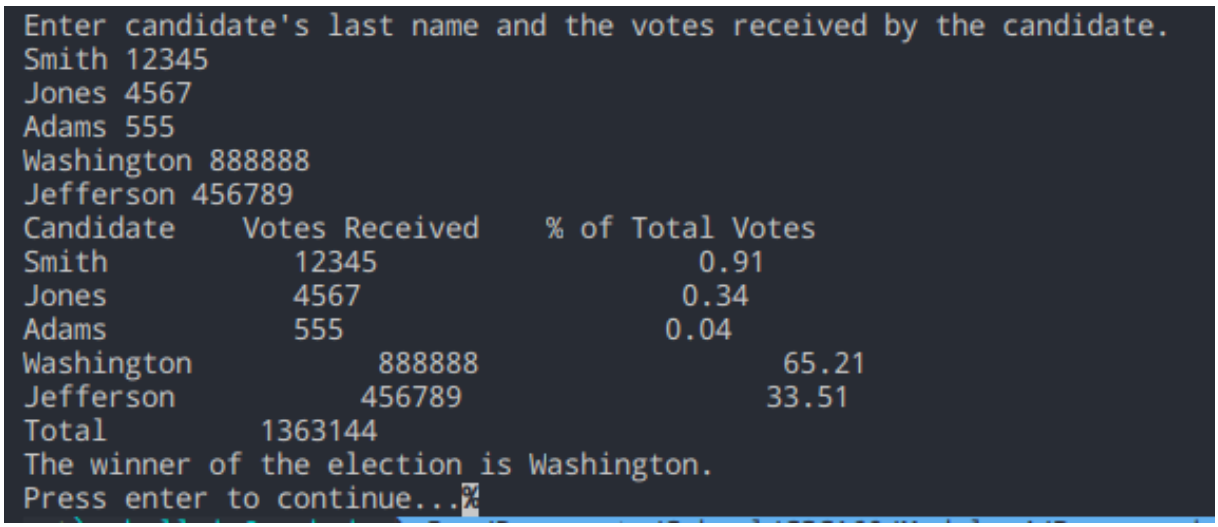
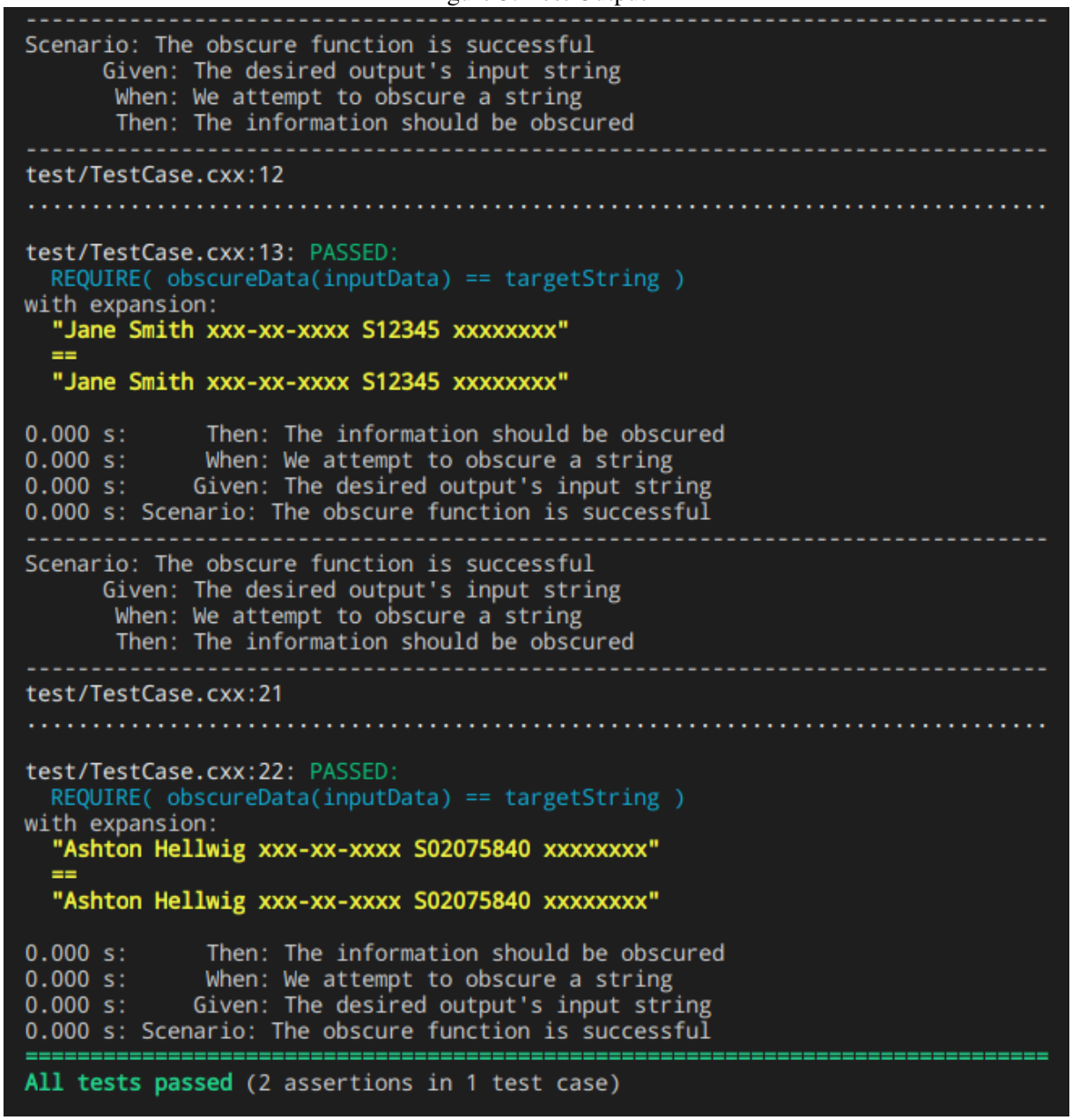



Figure 2: Running Chapter 10‘s Program

## B Unit Tests

Tests were written with the Catch2 library. The output is shown below.

Figure 3: Test Output



### B.1 Unit Test File

Below I have included the code used to run the unit test for reference.

```
1  // 00_CatchMain.cxx
2
3  // In a Catch project with multiple files, dedicate one file to compile the
4  // source code of Catch itself and reuse the resulting object file for linking.
5
6  // Let Catch provide main():
7  #define CATCH_CONFIG_MAIN
8
9  #include "../include/catch2/catch.hh"
10
11 // That's it
12
13 // Compile implementation of Catch for use with files that do contain tests:
14 // - g++ -std=c++11 -Wall -I$(CATCH_SINGLE_INCLUDE) -c 000-CatchMain.cpp
15 // - cl -EHsc -I%CATCH_SINGLE_INCLUDE% -c 000-CatchMain.cpp
```

Listing 3: 00_CatchMain.cxx

```
1  // main() provided in 00_CatchMain.cxx
2
3  #include "../include/book.hh" // TBD
4  #include "../include/catch2/catch.hh"
5  #include "../include/options.hh" // Argument parsing
6  #include <fstream>               // std::fstream
7  #include <iostream>              // std::cout
8  #include <string>                // std::string
9
10 class DBConnection {
11 public:
12   static DBConnection createConnection(std::string const & /*dbName*/) {
13     return DBConnection();
14   }
15
16   bool executeSQL(std::string const & /*query*/, int const /*id*/,
17                   std::string const &arg) {
18     if (arg.length() == 0) {
19       throw std::logic_error("empty SQL query argument");
20     }
21     return true; // ok
22   }
23 };
24
25 class UniqueTestsFixture {
26 protected:
27   UniqueTestsFixture() : conn(DBConnection::createConnection("myDB")) {}
28
29   int getID() { return ++uniqueID; }
30
```

```
31 protected:
32   DBConnection conn;
33
34 private:
35   static int uniqueID;
36 };
37
38 int UniqueTestsFixture::uniqueID = 0;
39
40 TEST_CASE_METHOD(UniqueTestsFixture, "Create Employee/No Name", "[
       create]") {
41   REQUIRE_THROWS(conn.executeSQL(
42       "INSERT INTO employee (id, name) VALUES (?, ?)", getID(), ""));
43 }
44
45 TEST_CASE_METHOD(UniqueTestsFixture, "Create Employee/Normal", "[
       create]") {
46   REQUIRE(conn.executeSQL("INSERT INTO employee (id, name) VALUES (?,
         ?)",
47                           getID(), "Joe Bloggs"));
48 }
49
50 // Compile & run:
51 // - g++ -std=c++2a -Wall -Iinclude -isystem include/catch2 -isystem
52 // include/cxxopts -o out/bin/test.bin test/01_TestStudentClass.cxx
53 // 000-CatchMain.o && 110-Fix-ClassFixture
54 // --success
55 // - cl -EHsc -I%CATCH_SINGLE_INCLUDE% -Iinclude 110-Fix-ClassFixture
       .cpp
56 // 000-CatchMain.obj && 110-Fix-ClassFixture --success
```

Listing 4: 01_TestBookType.cxx