Name: Austin Riha
Course: Procedural Programming (ELEN/COSC 1310)
Date: January 23, 2026

Homework: Chapters 1 and 2

Problem 1: Basic Structure

Task: Output Name and Course

Write a C program that prints your name on one line and your course name on the next line.

Source Code

```c
#include <stdio.h>

int main() {
    printf("Austin Riha\n");
    printf("Procedural Programming: ELEN/COSC 1310\n");
    return 0;
}
```

Output

```
[Running] cd "c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\" && gcc Temp.c -o Temp &&
"c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\"Temp
Austin Riha
Procedural Programming: ELEN/COSC 1310

[Done] exited with code=0 in 0.54 seconds
```

Problem 2: Multiple Includes

Task: Include Math Library

Write a program that includes `<stdio.h>` and `<math.h>`, then prints "Math library included successfully!"

Source Code

```c
#include <stdio.h>
#include <math.h>

int main() {
    printf("Math Library included successfully!\n");
    return 0;
}
```

Output

```
[Running] cd "c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\" &&
gcc Temp.c -o Temp &&
"c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\"Temp
Math Library included successfully!

[Done] exited with code=0 in 0.575 seconds
```

Problem 3: Comments Practice

Task: Comment Types

Write a program with a multi-line header comment, two single-line comments, and a specific printf statement.

Source Code

```c
#include <stdio.h>
#include <math.h>

int main() {
    /* This is a comment
       and this is another comment
       and one more just for fun */

    // Single line comment
    // Guess what? More comments! :)

    printf("Comments are useful!\n");
    return 0;
}
```

Output

```
[Running] cd "c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\" &&
gcc Temp.c -o Temp &&
"c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\"Temp
Comments are useful!

[Done] exited with code=0 in 0.619 seconds
```

Problem 4: Program Documentation

Task: Student Grade Calculator

Create a well-documented C program for a "Student Grade Calculator" with a complete header and explanatory comments.

Source Code

```c
/*
 * Program Name: Student Grade Calculator
 * Author: Austin Riha
 * Date: 01/22/2026
 * Description:
 * This program initializes the Student Grade Calculator
 * and prints a welcome message to the user.
 */

#include <stdio.h>

int main() {
    // Print the welcome message to the console
    printf("Welcome to the Student Grade Calculator\n");

    return 0; // Return 0 to indicate successful execution
}
```

Output

```
[Running] cd "c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\" &&
gcc Temp.c -o Temp &&
"c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\"Temp
Welcome to the Student Grade Calculator

[Done] exited with code=0 in 0.567 seconds
```

Problem 5: Minimal vs. Complete

Task: Comparison

Write the minimal valid C program, then a complete structured version. Compare the two approaches.

Code Comparison

Part 1: Minimal Valid Program

```c
int main(){}
```

Part 2: Complete Structured Program

```c
/*
 * Program Name: HW2
 * Author: Austin Riha
 * Date: 1/22/26
 * Description: A basic program that shows proper documentation.
 */

#include <stdio.h>

int main(void) {
    // prints confirmation to the console
    printf("This is what proper documentation looks like!\n");
    return 0; // returns 0 to show that the code was successful
}
```

Answer

Difference & Learning Value: The primary difference is code density and readability. The minimal program is syntactically valid and efficient for low-memory environments, but it relies on compiler defaults.

The structured program is significantly better for learning. It allows the user to read, understand, and follow the logic without guesswork. The comments act as a guide, preventing the learner from struggling to understand the "why" behind the syntax.

Problem 6: Loop Output

Task: Loop Analysis

Analyze the provided loop code.

Source Code

```c
#include <stdio.h>
int x, y;
int main(void) {
    for(x=0; x<10; x++, printf("\n"))
        for(y=0; y<10; y++)
            printf("X");
    return 0;
}
```

Output

```
[Running] cd "c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\" &&
gcc Temp.c -o Temp &&
"c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\"Temp
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX

[Done] exited with code=0 in 0.512 seconds
```

Answer

What does the program do?

It prints a 10×10 grid of X's. The outer loop handles the newlines, while the inner loop prints the characters.

Problem 7: Character Loop

Task: Code Analysis: ex02-04.c

Analyze the provided character loop code (`ex02-04.c`).

Source Code

```c
/* ex02-04.c */
#include <stdio.h>
int main(void) {
    int ctr;
    for (ctr = 65; ctr < 91; ctr++) {
        printf("%c", ctr);
    }
    printf("\n");
    return 0;
}
```

Output

```
[Running] cd "c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\" &&
gcc Temp.c -o Temp &&
"c:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming\"Temp
ABCDEFGHIJKLMNOPQRSTUVWXYZ

[Done] exited with code=0 in 0.6 seconds
```

Answer

What does the program do?

It prints the ASCII characters from A to Z (ASCII values 65 through 90) on a single line.

Problem 8: String Input

Task: Code Analysis: ex02-05.c

Analyze the provided string input code (`ex02-05.c`).

Source Code

```c
/* ex02-05.c */
#include <stdio.h>
#include <string.h>

int main(void) {
    char buffer[256];
    printf("Enter your name and press <Enter>:\n");
    fgets(buffer, 42, stdin);
    printf("\n Your name has %ld characters and spaces!", strlen(buffer));
    return 0;
}
```

Output

```
PS C:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming> gcc Temp.c -o
 Temp.exe
PS C:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming> ./Temp.exe
Enter your name and press <Enter>:
Austin

 Your name has 7 characters and spaces!
PS C:\Users\austi\my-heroui-app\COSC-1310_Procedural_Programming>
```

Answer

What does the program do?

It prompts the user for their name, reads the input using `fgets`, and then counts and prints the total number of characters (including spaces) in the name.

Problem 9: Error Identification

Task: Debugging

Identify the errors in the provided code snippet.

```
1  include <stdio.h>
2
3  int main() {
4      printf("Hello World\n")
5      return 0;
6  }
```

Identified Errors

Line	Error Type	Correction
1	Preprocessor Directive	Missing `#` before `include`
4	Syntax Error	Missing semicolon `;` at end of line

Problem 10: Program Structure Analysis

Task: Analysis Questions

Analyze the provided "Program Analysis Exercise" code.

```c
#include <stdio.h>
#include<stdlib.h>
int main() {
    printf("Program Analysis Exercise\n");
    printf("Header files are important!\n");
 return 0;
}
```

Answer

1. **What header files are included?**
   `stdio.h` and `stdlib.h`

2. **What does the program output?**
   The program outputs two lines:

   ```
   Program Analysis Exercise
   Header files are important!
   ```

3. **What would happen if you remove `#include <stdio.h>`?**
   The compiler will issue a warning (or error) because `printf` is undefined.