

# COP 4338 Assignment 4: Space Mission Control Autograders - Testing Guide

## Execution Permissions Notice

If you **do not** have execute permissions for instructor-provided framework files, you can manually assign the necessary permissions to specific files or to all files within a directory using the following command on a **Linux system**:

**Note:** When files are uploaded or transferred to Ocelot server (`ocelot-bbhatka1.aul.fiu.edu`), execute permissions **may be stripped** due to security restrictions. In such cases, you must explicitly grant **read (r)**, **write (w)**, and **execute (x)** permissions.

### [Granting Permissions to All Items in the Current Directory](#)

```

1 # Check current permissions
2 ls -l
3
4 # Grant read, write, and execute permissions (safe and preferred - applies only what's missing)
5 chmod +rwx *
6
7 # Assign full permissions (read, write, execute) to all users for all files in the current
8 # directory
9 # ⚠️ Not preferred - use only as a last resort!
10 chmod 777 *
11
12 # Verify that permissions were applied
13 ls -l

```

## How to Use the Provided Test Cases - *for manual testing*:

This section outlines the recommended process for manually validating your implementation against sample test cases.

### 1. Simple Test Case

1. **Copy the contents of `testcases_simple.txt` into `TESTCASES.txt`.**

2. **Run the instructor's sample executable to generate the expected output:**

⚠️ The file `TESTCASES.txt` must be present in the directory where `A3_sample` is located.

💡 The file `TESTCASES.txt` should contain the test cases you wish to run to generate the `EXPECTED_OUTPUT.txt`.

```
1 | ./A4_sample > EXPECTED_OUTPUT.txt
```

3. **Run your implementation (compiled via the provided `Makefile`) to generate your output:**

```
1 | ./space_mission > STUDENT_OUTPUT.txt
```

4. **Compare your output with the expected output:**

```
1 | diff STUDENT_OUTPUT.txt EXPECTED_OUTPUT.txt
```

 **No differences means a 100% match and your implementation is correct. Congratulations!**

## 2. Moderate and Rigorous Test Cases

- Repeat the steps above using the corresponding testcases:
  - Replace the contents of `TESTCASES.txt` with the contents of `testcase_moderate.txt` and `testcase_rigorous.txt`, respectively.
  - Regenerate the corresponding `EXPECTED_OUTPUT.txt` using `./A4_sample > EXPECTED_OUTPUT.txt`.
  - Rerun your implementation and perform the comparison: `diff STUDENT_OUTPUT.txt EXPECTED_OUTPUT.txt`.

 **Important:** The test cases provided (simple, moderate, rigorous) are designed to help verify the functional correctness of your solution. However, **instructor will use additional complex and comprehensive test cases** during grading. Therefore, it is **mandatory** that your implementation passes all three provided test cases—simple, moderate, and rigorous — **to maximize the likelihood that it will also pass the instructor's test cases during final grading.**

## How to use the autograders

The autograder scripts is available to facilitate automated testing throughout your development process **at any stage**. A correct implementation will earn **90 out of 100 points** through the autograders. The remaining **10 points** will be awarded based on:

- Adherence to submission guidelines
- Code structure and quality
- Code documentation

\* Please note that **instructor will use the same autograders for the final grading.**

## 1. Required Files and Directory Structure

Ensure that the following files are located in the same directory:

```
1 | space_mission/
2 |   mission_control.c           # Your implementation (Functions 1 & 2)
3 |   communication.c            # Your implementation (Function 3)
```

```

4 └── file_io.c                                # Your implementation (Functions 4 & 5)
5 └── memory_mgmt.c                            # Your implementation (Function 6)
6 └── Makefile                                  # Makefile to build application (⚠ DO NOT MODIFY)
7 └── space_mission.h                          # Header file (⚠ DO NOT MODIFY)
8 └── driver.c                                 # Application driver (⚠ DO NOT MODIFY)
9 └── TESTCASES.txt                           # Test input data
10 └── EXPECTED_OUTPUT.txt                     # Expected output
11 └── autograder_space_mission_system.sh    # Autograder
12 └── batchgrader_space_mission_system.sh   # Batch autograder
13 └── simple_missions_data.txt               # Mission data files (⚠ DO NOT MODIFY)
14 └── moderate_missions_data.txt            # Mission data files (⚠ DO NOT MODIFY)
15 └── rigorous_missions_data.txt           # Mission data files (⚠ DO NOT MODIFY)
16 └── empty_missions_data.txt              # Mission data files (⚠ DO NOT MODIFY)
17 └── invalid_missions_data.txt            # Mission data files (⚠ DO NOT MODIFY)
18 └── Harry_Potter.ZIP                      # Your final submission ZIP folder

```

## 2. Executing the Autograder

Run the autograder script using the following command:

```

1 # Run autograder
2 ./autograder_space_mission_system.sh

```

This script will compile your code, run the test cases, and provide a detailed score report.

## 3. Executing the Batch Autograder

**⚠ Remember:** Your final submission **ZIP folder** must be in your current working directory to run the batch autograder.

Run the batch autograder script using the following command:

```

1 # Run batch autograder
2 ./batchgrader_space_mission_system.sh

```

This script will compile your code, run the test case, provide a detailed score report, and **flags error for any missing required files**.