📚 COP 4610 Assignment 1: Kernel Data Structures - Testing Guide

🔄 How to Use the Provided Test Cases - *for manual tesing*:

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***:

```
./A1_sample > EXPECTED_OUTPUT.txt
```

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

```
./kernelDS > STUDENT_OUTPUT.txt
```

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

```
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 `testcases_moderate.txt` and `testcases_rigorous.txt`, respectively.
  - Regenerate the corresponding `EXPECTED_OUTPUT.txt` using `./A1_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 autograder

The autograder script 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 autograder. 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 autograder for the final grading.**

1. Required Files and Directory Structure

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

```
Assignment_1/
├── ds_header.h            # Header file (provided - DO NOT MODIFY)
├── driver.c               # Application driver (provided - DO NOT MODIFY)
├── Makefile               # Builds your application (provided - DO NOT MODIFY)
├── autograder_kernelDS.sh # Autograder (provided - DO NOT MODIFY)
├── TESTCASES.txt          # Test cases (copy simple/moderate/rigorous testcases here)
├── EXPECTED_OUTPUT.txt    # Expected results (generated by executing ./A1_sample > EXPECTED_OUTPUT.txt)
├── stack.c                # Student implementation
├── circular_queue.c       # Student implementation
├── circular_linked_list.c # Student implementation
├── min_heap.c             # Student implementation
└── bitmap.c               # Student implementation
```

2. Executing the Autograder

Run the autograder script using the following command:

```
# Run autograder
./autograder_kernelDS.sh
```

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