

# CS 361 - Homework 06

Professor Mark W. Boady

## 1 Overview

In this homework, you will work with semaphores, locks and/or condition variables to solve another resource management problem. This is another problem inspired by Operating Systems and determining how to allow many independent threads to use the same resources.

## 2 Trapped on an Island

There is an island with  $a$  adults and  $c$  children trapped on it. There is exactly one boat available. The job of your program is to get all the people off the island and back to the mainland.

Your program will take two command line inputs. Both must be integers and strictly larger than 0. The first number is how many adults are on the island. The second input is how many children are on the island. Your program should produce an error if no value is given or an invalid input is given.

For example, to run with 5 adults and 3 children we would run `./bin/island 5 3`.

Each adult and each child **must** be implemented as an individual thread. Once a thread is on the mainland and is no longer needed, it may exit.

The shared resource between the threads is the boat. There is only one boat available. It has two seats, a driver and a passenger. The driver may row the boat from the island to the mainland. The boat has the following requirements:

- It has exactly two seats. One driver (who rows the boat) and one passenger who just rides in the boat.
- The boat is small and has a weight limit. It cannot hold two full grown adults.
  - 1 child or adult can ride in the boat alone (passenger seat empty)
  - 2 children can ride in the boat
  - 1 child and 1 adult can ride in the boat
  - 2 adults **cannot** ride in the boat
  - It takes between 1-4 seconds (random) to row from the island to the mainland or vice versa
- No thread may row the boat more than 4 times without a break. (Sitting in the passenger seat for 1 direction or on the island counts as a break.)

You need to make sure to always return the boat to the island for the next group. If the boat is not returned to the island, all remaining threads will be trapped forever. This means that if two threads take the boat to the mainland, you cannot have both exit or the boat will never be returned.

Once all the threads are safely off the island, print out the following statistics about your program.

1. Number of times the boat traveled to the mainland
2. Number of times the boat returned to the island
3. Number of boats with 2 children
4. Number of boats with 1 child and 1 adult

5. Number of boats with only 1 person (child or adult)
6. Number of times adults were the driver
7. Number of times children were the driver

You **may** use the semaphore code provided in lecture. Do not use the starvation free mutex code from lecture, it will not help you. You **may** use the C++ standard classes like mutex locks and conditional variables. You may use these directly or implement them as part of one or more classes.

You may use the basic rand command or use the C++11 non-deterministic hardware entropy based random number generator.

You **must** make sure that your I/O and any shared data structures are thread safe.

You **must** ensure the following:

- All threads make it to the mainland
- The boat never contains an incorrect number of people

Once you have completed your code, you will run a few tests and write your thoughts on how your code works. You will write your thoughts in a readme file.

## 2.1 Output Style

When a thread gets in the boat print out what has happened.

Template: [Thread Name] got into the [position] seat of the boat.

For example, if a child got into the driver seat and an adult got into the passenger seat we print

```
1 Child 1 got into the driver's seat of the boat.
2 Adult 4 got into the passenger seat of the boat.
```

When that boat is being used print out what happens.

Template: Boat is traveling from [start] to [stop].

For example, if the boat was going from the island to the mainland we print

```
1 Boat is traveling from island to mainland.
```

Once the all threads have made it to the mainland, print out a final summary using the following template. (Numbers below are nonsense, they are just provided for formatting. We don't want to give any hints about the design by providing real results.)

```
1 Summary of Events
2 Boat traveled to the mainland: 12
3 Boat returned to the island: 11
4 Boats with 2 children: 7
5 Boats with 1 child and 1 adult: 3
6 Boats with only 1 person (child or adult): 4
7 Times adults where the driver: 4
8 Times children where the driver: 11
```

## 2.2 Implementation

You are expected to write professional code. Use good variable and function names. Provide comments explaining how the code works. Documents each function in the header file with comments. You will be graded on style as well as functionality.

## 2.3 Citations

If you use any outside resources, talk about algorithm design with other students, or get help on assignments you **must** cite your resources in the comments. Uncited sources are an Academic Honesty Violation. Cited sources may lead to a deduction depending on the amount of code used, but will not violate Academic Honesty Policies.

You are expect to write the majority of the code yourself and use resources for things like looking up commands. For example, if you forgot how to test if a file can be opened for reading you could look it up and cite a source. If you copy a critical algorithm and cite the code, you may still get a deduction for not developing the code yourself.

## 2.4 Readme

Your readme.md will include both instructions and reflections on your code. It must be stored on the root of your folder structure. It must use markdown formatting and be named readme.md.

There is no minimum or maximum length for the short essay questions, you are graded entirely on content. A short but comprehensive answer is better than a long confusing answer.

1. Your name and drexel ID (abc123@drexel.edu)
2. Instructions to run you code.
3. Short Essay Question 1: What did you use to protect the boat and why?
4. Short Essay Question 2: How did threads decide what position to take in the boat?
5. Short Essay Question 3: How did you reset the boat for the next group?
6. Short Essay Question 4: Why are you **certain** everyone will get off the island?
7. Short Essay Question 5: What was the most challenging part of this assignment?

## 2.5 Makefile

You **must** provide a makefile to compile your code. We will type make and it **must** build your program. We will the type make run and it must test your program. If there are any compile errors or a makefile is not provided we cannot test your code.

You must have the following make targets:

1. make - Builds the Program
2. make run - Runs a simulation with 7 adult and 9 children
3. make clean - Remove compiled code

## 2.6 Other Requirements

If you submission does not meet the following guidelines we will not be able to grade it.

1. You **must** use the C++ 17 Standard threads. No other thread libraries (pthreads, boost, etc) may be used. <https://en.cppreference.com/w/cpp/header/thread>
2. Code **must** run on tux and be compiled with g++.
3. All code **must** compile using the C++ 17 or above standard. (`--std=c++17`)
4. All code **must** be submitted to blackboard learn in a zip file.
5. A working makefile **must** be provided.

6. Must provide a readme file
7. You may use libraries in the C++ 17 standard unless noted elsewhere in the instructions. <https://en.cppreference.com/w/cpp/header>
8. Your code **must** compile. You should always submit stable code, we will not debug code that does not compile.

### 3 Grading

This homework is worth 100 points.

Question 1 : 9 points

Main Program

- (a) (2 points) Handles Command Line Arguments
- (b) (2 points) Starts all threads
- (c) (2 points) Joins all threads before exiting
- (d) (3 points) Overall Style

Question 2 : 70 points

Threads

- (a) (25 points) Adult Thread Logic
- (b) (25 points) Child Thread Logic
- (c) (20 points) Boat Logic

Question 3 : 17 points

Readme.md

- (a) (1 point) Name and Email —
- (b) (1 point) Instructions
- (c) (3 points) Short Essay 1
- (d) (3 points) Short Essay 2
- (e) (3 points) Short Essay 3
- (f) (3 points) Short Essay 4
- (g) (3 points) Short Essay 5

Question 4 : 4 points

Instructions

- (a) (2 points) Makefile is correct
- (b) (2 points) Required Files Submitted