Project 1 Report

CMPS 455

Spring 2014

Marcus Amos

Task 1

1. Why is the ability to check input so important?

The ability to check input is a primary method of ensuring your program works properly without errors. Invalid input can cause even properly coded functions to fail, which can result in minor or major problems in program execution.

1. Other than simply providing the wrong type of input, what other ways can you think of for bad input to cause an error? Consider situations other than typing input when prompted.

If functions are passing variables as parameters to other functions with input that is outside of the scope of the program, many programs can be caused to crash. That one bad input can have a domino effect and cause bad data to be stored and cause other bad input to be created for subsequent functions.

Task 2

1. Remove the busy waiting loop used whenever a thread shouts and run the task with 5 shouters and 5 shouts per shouter. Then have each thread yield once after shouting and run another test with the same parameters. Note your results and explain your observations.

Undo any changes made to accommodate this question before submitting your assignment.

When there were no busy wait loops, the process took 1060 ticks and there were multiple times when the same shouter got back to back shouts.

With a single busy wait loop, the same process took 1350 ticks and there were no back to back shouts from any shouter.

1. Temporarily disable your input validation, run a minimum of 5 tests with garbage input, and note the results. How would an end user react to this? Undo any changes made to accommodate this question before submitting your assignment.

Whenever characters are entered as either the number of shouters or the number of shouts, an infinite loop results. Whenever a character is entered when choosing the task, the program terminates. Whenever a negative numbered is entered, the program terminates.

Task 3

1. What other solutions can you think of to improper input on the command line?

The problem can be solved using more brute force method of only if-else statements.

Task 4

1. In your own words, explain how you implemented each task. Did you encounter any bugs? If so, how did you fix them? If you failed to complete any tasks, list them here and briefly explain why.

For task 2, I used a struct and typecasting to pass the data to the function. I then used the random number generator to randomly pick with thread to run next. If thread 1 randomly chose a number that was not 1, the problem would call currentThread->Yield(). I then called a while loop to randomly shout output until the limit was reached. I learned how to used the random number generator in Nachos

1. What did you learn from working on this assignment?
   1. How to use the random number generator in Nachos
   2. How to fork multiple threads
   3. How to simulate busy-wait loops
   4. How to interpret command line functions
   5. How to thoroughly test input for validity