# CS563 Assignment 4: Programming with Go

Instructor: Xinghui Zhao

Due: 11:59pm, February 16, 2020

## 1 Overview

Communicating sequential processes (CSP) is a formal language for describing patterns of interaction in concurrent systems. It is a member of the family of mathematical theories of concurrency known as process algebras, or process calculi, based on message passing via channels. CSP was highly influential in the design of a number of programming languages, including Go. The purpose of this assignment is to give you some practice on programming using Go.



## 2 Merge

In Lecture 4 (message passing), we discussed a filter process which merges two input streams. The pseudo code is shown in Figure 1. If we create a number of merge processes, and organize them in a tree structure as shown in Figure 2, we can easily sort a data stream.

The pseudo code uses the traditional message passing syntax. Here, your task is to implement this algorithm using Go.

## 3 Sieve of Eratosthenes

We have discussed Sieve of Eratosthenes algorithm for generating prime numbers in Lecture 6 (CSP). This algorithm uses filer communication pattern. Figure 3 shows the algorithm.

Your task is to implement this algorithm using Go.

```
chan in1(int), in2(int), out(int);
process Merge {
  int v1, v2;
  receive in1(v1);  # get first two input values
  receive in2(v2);
  # send smaller value to output channel and repeat
  while (v1 != EOS and v2 != EOS) {
    if (v1 <= v2)
      { send out(v1); receive in1(v1); }
    else   # (v2 < v1)
      { send out(v2); receive in2(v2); }
  }
  # consume the rest of the non-empty input channel
  if (v1 == EOS)
    while (v2 != EOS)
      { send out(v2); receive in2(v2); }
  else   # (v2 == EOS)
    while (v1 != EOS)
      { send out(v1); receive in1(v1); }
  # append a sentinel to the output channel
  send out(EOS);
}
```

Figure 1: Merge Pseudo Code



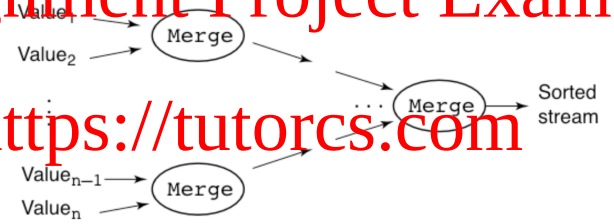






Figure 2: Merge Processes

```
process Sieve[1] {
  int p = 2;
  for [i = 3 to n by 2]
    Sieve[2]!i;   # pass odd numbers to Sieve[2]
}
process Sieve[i = 2 to L] {
  int p, next;
  Sieve[i-1]?p;               # p is a prime
  do Sieve[i-1]?next ->       # receive next candidate
    if (next mod p) != 0 ->   # if it might be prime,
      Sieve[i+1]!next;        #   pass it on
    fi
  od
}
```

Figure 3: Sieve of Eratosthenes

## 4 One-Lane Bridge

Cars coming from the north and south arrive at a one-lane bridge. Cars heading in the same direction can across the bridge at the same time, but cars heading in opposite directions cannot.

Develop a Go program to simulate the use of the bridge. Assume the cars are client processes, and the bridge is the server process. Have cars repeatedly try to cross the bridge. They should spend a random amount of time on the bridge and should delay for a random amount of time before crossing again. Print a trace of the key events in your simulation.

## 5 Submission

Submit the following on Blackboard:

1. Source code for the three applications
2. Sample output for each of the applications (screenshots in png, pdf, jpeg, etc.)



## 6 Grading Scheme

This assignment will be graded out of 100. For your information, the grading scheme is shown in the following table. Note that for this assignment, code readability is evaluated in the grading process, so please include comments in the source code when necessary.

| Item | Percentage |
|---|---|
| Merge | 30% |
| Sieve | 30% |
| One-lane bridge | 30% |
| Code readability | 10% |