Programming Assignment #1

COEN 296 Introduction to Cloud Computing
Department of Computer Engineering
Santa Clara University

Dr. Ming-Hwa Wang — Summer Quarter 2014
Phone: (408) 525-2564 — Email address: m1wang@scu.edu
Course website: http://www.cse.scu.edu/~mwang2/cloud/
Office Hours: Tuesday & Thursday 9:00-9:30pm

Due date: Midnight July 6, 2014

Message Driven Computing (MDC) (200 points)

There are two ways to do parallel programming. The first one is using a parallel compiler which automatically (or with limited help by programmer) translates sequential codes into parallel codes. The second one is the programmer has to write explicit parallel codes.

In order to write parallel codes, we need parallel programming languages. A parallel programming language has all sequential constructs of traditional sequential languages (e.g., C/C++, Java, etc.). Besides, a parallel programming language also contains parallel constructs.

There are several parallel languages and models. Message Driven Computing (MDC) is one of them. The basic ideas behind MDC are asynchronous message passing and pattern matching (similar to Actor). In MDC, there are many loci (the plural of locus). Each locus contains portal(s) and codes. Messages can be sent from one locus to any portal(s) of any locus or loci. Messages have types which are the same as portals. At each locus, if the arrived messages match certain pattern, then the code is executed and the matched messages are removed. During execution, more messages can be sent out and they may cause more computation. When all message received and no pattern can be matched at all loci, the program terminated. E.g.,

```
L1 ? P11(bool bMin), P12(int i1), P12(int i2) {
    int iResult =
        bMin ? i1 < i2 ? i1 : i2 : i1 < i2 ? i2 : i1;
    ! P21(bMin, iResult) @ L2;
}
L2 ? P21(bool bMin, int iResult) {
    if ( bMin ) {
        cout << "The minimum is " << iResult << endl;
    } else {
        cout << "The maximum is " << iResult << endl;
    }
}
```

Messages can optionally with delay. Write a general program which can implement the MDC efficiently using C/C++ or Java. The input example is:

```
// Note that the locus/portal name can be any identifier
// send messages to trigger the computation
// the #<num> means delay
! P11@L1 #1;
! P12(1)@L1 #2;  ! P12(5)@L1;
// pattern matching
// "+" means more than one, "*" with more than zero
L1 ? P11 P12+(int i) with_same i {
    int sum = 0;
    for (int k = 0; k < P12.length(); k++) {
        sum += P12[k].i;
    }
    ! P21@L2(sum) # 5;
}
L2 ? P21(int sum) {
    cout << "Total of " << sum << endl;
}
```

Your output should display:

```
// L1 ? P11  P12  P12 is executed once at time 2
// L2 ? P21 is executed once at time 7
// Program terminated at time 7
Total of 6
```

Student Name:

SSN/ID:

Score:
Correctness and boundary condition (60%):
Whitespace and free format compliance (5%):
Compiling without warning/error (5%):
Error Handling (5%):
Modular design, file/directory organizing, showing input, documentation, coding standards (20%):
Automation (5%):

Subtotal:
Late penalty (20% per day):
Special service penalty (5%):

Total score: