Name: ________________________

Student #: ____________________

Simon Fraser University

ENSC 351 –Craig Scratchley

351 Sample Midterm Exam

Enter “D100” on the computer answer sheet under “SECTION” and “0001” under “SPECIAL CODE”. When you are finished with the exam, hand in these exam pages, the dataServer.c code, as well as the computer sheet.

Please refer to the distributed code “dataServer.c” for questions A, B, and 01 through 10. This code is ***based*** on the time1.c code from the Krten textbook, but has significant differences. There are two written questions (marks as specified) and 12 multiple-choice questions (1 mark each), for a total of 20 marks.

A. Consider a single-threaded client that has a loop in which it will request data from the server, and when it gets data it will write the data out to standard output before going back to the top of the loop. Write the loop with the assumption that `coid` is the connection ID of the client’s connection to the server’s channel. (3 marks)

B. Draw using the collaboration graph notation the server, one data-requesting client, and two data-supplying threads (you may assume each is in a separate process if you wish). Indicate send blocking or reply blocking if either is to be expected. (5 marks)

Answer each of the following multiple-choice questions. Circle your choice for each question on these pages and eventually fill in the computer answer sheets.

01. In function *main()*, the line "exit (EXIT_FAILURE)" does what?

a. Ends the primary thread and then immediately ends the process.
b. Ends the process but the primary thread still remains.
c. Ends the primary thread but the process still remains.
d. Calls the *exit()* function after which the program goes on to the *receiveMessages()* function.
e. Exits the block of code associated with the "if" statement.

For questions 2 through 10… In function *main()*, consider if the line

```
pthread_create (NULL, NULL, receiveMessages, NULL);
```

was put before the line

```
receiveMessages (NULL);
```

(i.e. "uncomment" the *pthread_create()* line)

02. How many threads would now be in the process:

a. 0
b. 1
c. 2
d. 3
e. more than 3

03. Which variable and/or data structures would **have** to be protected from concurrent access to ensure correct operation of the program:

a. msg
b. clients[0], clients[1], …, clients[MAX_CLIENT – 1]
c. chid
d. both b and c
e. none of the above

04. Of the following choices, with QNX what would be the most-efficient way to protect any variables or data structures needing protection:

a. use one or more mutexes
b. use one or more semaphores
c. use one or more barriers
d. use message passing
e. No variables or data structures need to be protected

05. If a semaphore named `sem4` and initialized to 1 was being used in the program to protect the consistency of one or more variables and/or data structures, what modified code would allow correct operation of these lines from function *gotAMessage()*:

```
for (i = 0; i < MAX_CLIENT; i++) {

    if (!clients [i].in_use) {

        // found one -- mark as in use, save rcvid
        clients [i].in_use = 1;
        clients [i].rcvid = rcvid;
        return;
    }
}
```

a.

```
sem_wait(&sem4);
for (i = 0; i < MAX_CLIENT; i++) {
    if (!clients [i].in_use) {
        clients [i].in_use = 1;
        clients [i].rcvid = rcvid;
        return;
    }
}
sem_post(&sem4);
```

b.

```
for (i = 0; i < MAX_CLIENT; i++) {
    sem_post(&sem4);
    if (!clients [i].in_use) {
        clients [i].in_use = 1;
        clients [i].rcvid = rcvid;
        return;
    }
    sem_wait(&sem4);
}
```

c.

```
for (i = 0; i < MAX_CLIENT; i++) {
    sem_post(&sem4);
    if (!clients [i].in_use) {
        clients [i].in_use = 1;
        clients [i].rcvid = rcvid;
        sem_wait(&sem4);
        return;
    }
    sem_wait(&sem4);
}
```

d.

```
for (i = 0; i < MAX_CLIENT; i++) {
    sem_wait(&sem4);
    if (!clients [i].in_use) {
        clients [i].in_use = 1;
        clients [i].rcvid = rcvid;
        sem_post(&sem4);
        return;
    }
    sem_post(&sem4);
}
```

e.

```
for (i = 0; i < MAX_CLIENT; i++) {
    sem_wait(&sem4);
    if (!clients [i].in_use) {
        sem_wait(&sem4);
        clients [i].in_use = 1;
        clients [i].rcvid = rcvid;
        sem_post(&sem4);
        return;
    }
    sem_post(&sem4);
}
```

06. Comparing the program resulting from a correct modification from question 5 (and any other needed modifications) with the original program (i.e. the program exactly as was given out with this exam), on average how much time would transpire using the modified program between receiving a synchronous (i.e. non-pulse) message and replying to it? Assume the programs are being run on a **uniprocessor** system.
   a. roughly half the time it would take the original program
   b. a little less time than it would take the original program
   c. exactly the same time it would take the original program
   d. a little more time than it would take original program
   e. roughly twice the time it would take the original program

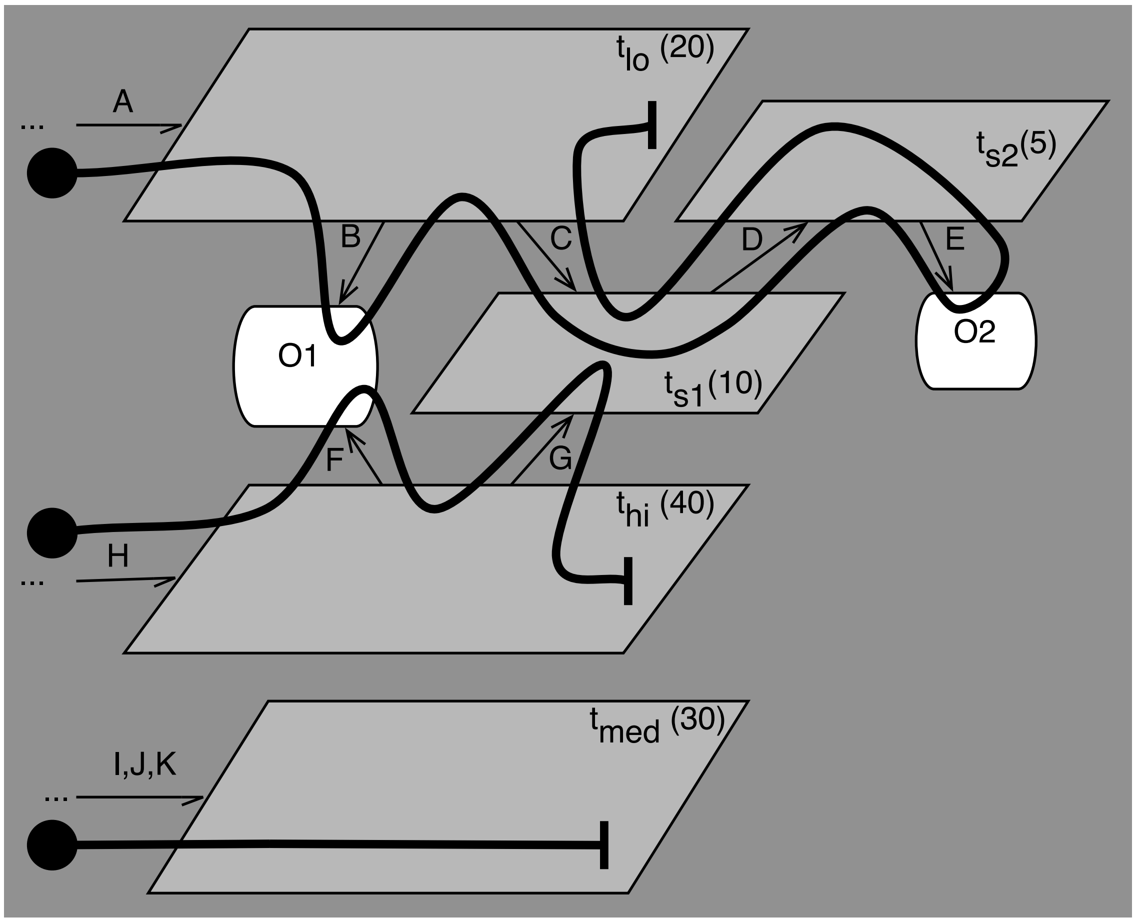
07. Using for both programs a **very lightly loaded 8-processor symmetric multiprocessor (SMP)** system, on average how much time would transpire using the modified program between receiving a synchronous message and replying to it?
   a. roughly 1/8 of the time it would take the original program
   b. roughly half the time it would take the original program
   c. roughly the same time it would take the original program
   d. roughly twice the time it would take the original program
   e. roughly 8 times the time it would take the original program

08. Considering FIFO and standard Round Robin scheduling for the resulting program …
   a. If the threads in the program use standard Round Robin scheduling, more context switching would typically result.
   b. The choice of FIFO or standard Round Robin would typically make no difference to this program.
   c. Round Robin scheduling would be the best choice for this program.
   d. both a and c are correct
   e. none of the above are correct

09. If we turn off priority inheritance and if the thread created by the `pthread_create()` call is given a lower priority than the primary thread …
   a. on a uniprocessor system the created thread would never be given the chance to start running
   b. on a lightly loaded symmetric multiprocessor system the created thread would never be given the chance to start running
   c. the created thread would receive about half the pulses and messages on a lightly loaded multiprocessor
   d. both a and b are correct
   e. none of the above are correct

10. Considering *pthread_join(), pthread_barrier_init(),* and *pthread_barrier_wait()* …
   a. One or more *pthread_join()* calls are needed for this program to run properly.
   b. One or more *pthread_barrier_init()* calls are needed for this program to run properly.
   c. One or more *pthread_barrier_wait()* calls are needed for this program to run properly.
   d. both b and c are correct.
   e. none of the above are correct

Refer to the following diagram for the remaining questions.



11. Assuming default QNX priority inheritance behaviour and that the priority of threads when created are shown in parentheses, what priority will thread t_{s1} be running at immediately after the sequence A,B,C,I,H,F

   a. 10.
   b. 20.
   c. 30.
   d. 40.
   e. None of the above.

12. With the previous assumptions, what priority will thread t_{s1} be running at immediately after the sequence A,B,C,I,H,F,G

   a. 10.
   b. 20.
   c. 30.
   d. 40.
   e. None of the above.