**CSCI 4727 Homework 3**

**Due: Wednesday, October 27, 5:00 pm**

This assignment continues the code started in Homework 1. This version will use ***processes*** to model each processing station and ***pipes*** that allow the stations to hand off each product record to the next station, until done.

The stations will be represented by individual processes, created by calls to fork() (as discussed in lecture). A station is one process; the maximum number of stations, as noted above, is set at 5, so you’ll need to fork off exactly 5 processes, not counting the parent.

Each station will read from a Unix pipe; recall that this is a one-directional communication method. A process reads or writes from/to a pipe a product\_record struct that must be cast to/from a character pointer. The main program must declare the pipes’ data structure as global data, and it must initialize the pipes *before* forking the children. The pipe() function’s parameter is an array of two integers, used as the file descriptors for the read and write channels of the pipe. Recall that its prototype is int pipe(int fd[2]); In this program, a process reads from exactly one pipe, but may write to any other process’ pipe. For example, process 0 will read from its pipe (pipe #0), and will write to the pipe for process 1 (pipe #1). Given that explanation, it’s probably best to declare an array of pipes (i.e., a two dimensional array of int), one per process, plus one pipe for stages writing back to the parent.

The stations will process one product\_record at a time according to the following, totally made up, and unrealistic rules; unless otherwise noted, station #i reads a record from pipe #i and writes it to pipe #i+1. Each station #i should also set the record’s stations[i] field from 0 to 1 to show that this station processed this record.

Station 0: compute the tax amount, 5% of the amount ordered. If the order number is >= 1000, this is a special order that has no shipping and handling charge – bypass station 1 and send this record directly to station 2.

Station 1: compute the shipping and handling, $10 plus 1% of the amount ordered.

Station 2: compute the order total.

Station 3: compute and display the running total of all the orders seen so far.

Station 4: display the record to the screen, with the fields labeled, then send the record back to the parent.

Don’t over-think the code for these stations – each one should be pretty simple.

The parent process will create all of the pipes and fork off all of the child processes (stations). The parent is responsible for reading each the input file, storing the data product\_record, and writing the product\_record to the first station’s pipe (station 0), but it *does not display* the record on the screen – that’s station 4’s job. Station 4 will also write that the order’s record back to the parent – use a separate pipe for this purpose. The parent is then responsible for writing each product\_record to the output file. After the parent has received the last product\_record back from Station 4 and written it to the output file, the parent should write a special product record containing -1 as the product’s idNumber field to each station; when the station receives this “all done” flag record, it exits. Just before exiting, each station should display the number of product records it has processed with a message, such as “Station 3 processed 17 product records”. The stations may report different numbers (because of the Station 0 rules). The parent should also display the total number of orders processed.

**Deliverables** Drop the source code for hw3 in the appropriate dropbox. If your program is multiple files (strongly suggested, but not required) – put the files in a zip folder.