# CS 3423 Test 2

1. **[30]** Write a C **function** with prototype:

   ```
   int extractrecs(char *fdb, int num1, int num2, char *fout, int rsz);
   ```

   This function will extract a range of records starting with record **num1** and ending with record **num2** from the file whose pathname is given by **fdb**, writing the extracted records to the file whose pathname is given by **fout**. Note that the file **fdb** will shorter by the number of records extracted. You may assume that **fdb** contains the necessary records and that $num1 <= num2$. The output file, after execution, should only contain the extracted records.

   You may also assume that the file **fdb** consist of an integral number of records and that all records have the same size, **rsz**. A record is merely a consecutive block of bytes of size **rsz**. The record indexing starts at 0 so record 0 consists of bytes 0 through $rsz - 1$, record 1 consists of bytes $rsz$ through $2rsz - 1$, and so on. The function must both open and close the files and reads/writes **must** be record sized. If successful, **extractrecs()** will return a 0 else **extractrecs()** will return -1. The function must use low-level I/O except for printing error messages and you may assume that $rsz \leq 1024$.

2. **[25]** You have a condo in Q. Roo and want to be able to ssh to your wireless router (running openwrt) but the ip address keeps changing. So you simply have a cron job which uses netcat to send the string "condo" to the ssh port on your machine. This creates a log entry in `/var/log/messages` similar to

   ```
   Apr 20 13:15:01 Bad ssh protocol condo from 187.153.61.110
   ```

   You want a `single` program called `getcondo` which will scan the `messages` file and print out the date/time and the ip address of the condo. This can be done with the following command

   ```
   egrep condo /var/log/messages | sed 's/Bad ssh protocol condo from //'
   ```

   Write a single C program which will accomplish the same thing as the above command using the unix programs `egrep` and `sed`. Note that the single quotes in the `sed` command are there only to protect the string from the shell and are not seen by `sed`.

3. **[25]** Write a C program which will create a **total** of 33 processes sharing two pipes. Each process must be assigned a unique index 0, 1, ..., 32. The processes will be divided into two groups based on their pid, those with an even pid will be in the Even group and those with an odd pid in the Odd group. Each process in the Even group will write it's index and pid to the pipe of the Odd group while each process in the Odd group will write their index and pid to the pipe of the Even group. All processes will then read as many messages as possible from their group's read pipe printing on stdout, for each message read, a line similar to:

   ```
   Process 35894 index 24 heard from process 35991 index 17
   ```

   Be careful that your code doesn't hang and messages don't get mixed up!

(OVER)

4. **[25]** Write a `function`, called `xcmd()`, which will read all messages from a pipe. The message structure is given below and the read file descriptor of the pipe is passed to the function. Each message will contain the name of an executable and the names of two fifos. The function will cause the execution of the executable with no arguments and the stdin and stdout of the process should be connected to the input (data) FIFO, `dfifo`, and the output (return) FIFO, `rfifo`. The function `xcmd()` has a prototype of:
`int xcmd(int fd);`
and the message structure is:

```
struct command
  {
    char cmd[64];      /* Null-terminated name of executable          */
    char dfifo[64];    /* Name of data FIFO                           */
    char rfifo[64];    /* Name of return FIFO                         */
  };
```