CS 3423 Test 2

1. [**25**] Write a C **function** with prototype:

   ```
   int delrec(unsigned int num, char *fdb, unsigned int rsz);
   ```

   This function will delete record `num` from file whose name is given by `fdb`. Furthermore you may assume that the file 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 file and read/writes **must** be record sized. If successful, **delrec()** will return a 0 else **delrec()** will return -1. The function must use low-level I/O except for printing error messages and you may assume that $rsz \leq 1024$. **Note** that the file should be smaller!

2. [**25**] Write a **function**, called `execpipe`, that will be passed two structures of the form:

   ```
   struct command
     {
       char *cmd;      /* pathname of a command */
       char *argv[10]; /* Null terminated argv list of arguments  */
     };                /* (including the basename of command cmd) */
   ```

   The function `execpipe()` with prototype:
   `int execpipe(struct command cmd1, struct command cmd2);`
   will fork and exec both commands, piping the output of the first command into the second command. The function **returns** 0 on success and -1 on error. Notice that this is a **function** and the original process **expects** a return!

3. [**25**] Write a C `program` which will fork 30 children assigning each process a unique index from 0 to 30. There should be a common pipe shared by all the processes. The processes with an even index will write the numbers 1 to 1000, each number together with the processes pid, to the pipe. The odd processes (those with an odd index) will read as many messages as they can, adding up the numbers read (not the pids) and when the pipe is empty each odd process will write its pid and sum to stdout in the following format:

   ```
   Process 47836 total 14327
   ```

(OVER)

4. **[25]** Write a C `function` with prototype

```
int xprogs(int fd);
```

which will read **all** messages of varying size from the pipe whose file descriptor is passed to the function. A message consists of three parts, the first part is a header which contains the lengths of the next two parts,

```
struct header {
  int flen;    /* length of executable name including null terminator */
  int alen;    /* length of single argument including null terminator */
}
```

the second part, of length `flen`, is the null-terminated name of an executable which takes a single argument while the third part, of length `alen`, is a null-terminated argument string consisting of a single argument. The function will cause the execution of all executables with their single arguments and return -1 on error or 0 on reaching an end-of-file on the pipe.