CS-2413 Final

1. [15] Suppose a perl program, flurp, is invoked with the command:

```
flurp -x -vf file1 file2 file3 file4
```

file1 contains the word "dir4" and file3 contains only "dir5" and "dir6" file4 is empty

At the start of the program.

a) What is the list value of @ARGV?

b) What is the value of $#ARGV?

c) What value is used in the condition test for @ARGV:

```
while ( @ARGV ) {
```

d) If the following statements are executed first.

```
$a = pop;
$b = shift;
$c = shift;
$d = <>;
$e = <>;
```

i. What is the value of $a?

ii. What is the value of $e?

iii. What is the value of $ARGV?

e) On the other hand if the following statement were to be executed first,

```
$a = shift;
$b = shift;
$c = shift;
@d = <>;
```

i. What is the value of $b?

ii. What is the value of @d?

2. [20] Write a Perl script, pgrep, which will search all TEXT files under the directories given on the command line for the regular expression given as the first argument on the command line. Print out the pathname of each file in which a match is found. A pathname should be printed a maximum of once. A sample invocation:

```
pgrep '\w+ [mM]aynard' dir1 dir2 dir3
```

3. [20] Write a Perl script, called `inslines`, which will insert a block of lines from a first file after a given line in a second file. Thus to insert lines 15 through 25 in `infile` after line 45 in `outfile` the command would be:

```
inslines 15 25 infile 45 outfile
```

Notice that all lines of `outfile` are still in the file, there are merely the additional lines after [illegible].

4. [25] Write a perl script which will print out the minimum and maximum scores in each project in a cs2413 class. Assume that all of the project reports are kept in directories, `/usr/projects/project1, ..., /usr/projects/project7`. Each project report (if turned in) is named with the students login similar to:

```
/usr/projects/project4/maynard
```

and in this file is the total number of points in a line similar to the following:

```
Total Points:        28
```

There are only project report files in these directories and the output for the seven projects should be similar to:

Project	Min Score	Max Score
1	15	24
2	21	30
	.	.
.	.	.
7	5	30

5. [25] Write a C function with prototype:
`int delrec(unsigned int num, char *file, unsigned int rsz);` A file can be considered to be a sequence of record of a fixed size, say `rsz` where the first `rsz` bytes are to be considered record 0, the next `rsz` bytes are record 1. .... The function `delrec` will delete record number `num` in the file, `file`. Every record from record number `num+1` and greater must be moved forward one position. The function must both open and close the files. If successful, `delrec` will return a 0 else `delrec` will return -1. The function should use low-level I/O.

6. [25] Implement a function, ppconnect, with prototype:

```
int ppconnect(int fd[], const char *command);
```

which will create a process executing command and two pipes connecting to the stdin and stdout of the process giving the read end from stdout and write end of the appropriate pipe in fd[0] and the write end of the other pipe in fd[1]. Assume that you will use the current PATH to find command and that command has no arguments.

7. [25] You have a copy of a nis+ password file, named nispw.doc that you want to merge with the current copy of your nis+ password file throwing out the duplicate lines. Since this has to be done frequently you want to write a C-program to do this.

   If you do this by hand you would do:
   `niscat passwd.org_dir | sort -u - nispw.doc`
   and thus sort will sort the output of the niscat command whose output to stdout will be read by sort from stdin and the file nispw.doc printing the sorted output on stdout. Write a C function with prototype:
   `int mergepw();`
   which will print the sorted files on stdout returning -1 on error and 0 otherwise.

8. [25] Write a C program which will create 33 children processes all knowing about a single pipe. Each process (including parent) will write 10 messages to the pipe each consisting of the process' pid. Each of the processes will then read as many messages (one at a time) from the pipe as possible, printing out on stdout a message containing the process' pid and the message pid. The general form of the message is:
   `Process 12475 received pid 73134`
   Each process will exit when there are no more messages in the pipe.

9. [25] In the system that you are writing, many programs (called clients) will be executing and will need other programs, called servers, to process some data. In this version of the system each program will write to a well known fifo (named pipe) with the name "/tmp/dispatch". This fifo already exists in the file system. The dispatcher program reads each message from the fifo. execs the appropriate server after connecting the pipes. The server is expecting its data via stdin and will write its data to stdout. The dispatcher must be certain that the server's stdout is redirected to the return fifo named in the message and that the server's stdin is redirected to the data fifo named in the message. It is the clients responsibility to write the data to the data fifo. The format of the message is:

Data	Field (first byte - last byte)
program name (null terminated)	0-127
return fifo name (null terminated)	128-255
data fifo name (null terminated)	256-383

Note the dispatcher program for this system assuming that there may be several dispatcher programs executing at the same time.
Note that the dispatcher does not terminate but constantly reads messages from the dispatch pipe.