CSC 209H5 S 2018 Midterm Test
Duration — 45 minutes
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____ **First Name:** _____

Instructors: Dema, Petersen

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck!

This midterm consists of 4 questions on 10 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*

Comments are not required, although they may help us mark your answers.

No error checking is required except where specifically requested.

You do not need to provide include statements for your programs.

If you use any space for rough work, indicate clearly what you want marked.

1: _____/ 3
2: _____/ 2
3: _____/ 3
4: _____/ 7

TOTAL: _____/15

Question 1. [3 MARKS]

Bash

You have two executables, `encode` and `decode`. Both programs read from `stdin` and emit text to `stdout`. `encode` is intended to read data from a file and puts it into a specific format; `decode` takes data in that format and regenerates the original file. Both files exit with status code 0 on success and 1 on failure.

Part (a) [1 MARK]

Write the single-line command you would enter on the shell to execute `encode` on data from the file `grades.csv` and place the result in the file `grades.enc`.

Part (b) [1 MARK]

Assume the line in the previous question has just been run. Write bash code that prints "Success" if the encoding was successful and "Failure" otherwise.

Part (c) [1 MARK]

You've made changes to both executables and wish to commit those changes – but no other files – into an existing repository. Write the git commands required to get both files into the remote repository. (i.e., What you do to commit your lab files for grading each week?)

Question 2. [2 MARKS]

gcc and Make

Assume you have the files `encode.c`, `crypto.c`, and `crypto.h`. Both source files include the header file. Both source files are required to build the executable `encode`.

Write a `Makefile` that includes rules to build the executable `encode` in three steps by compiling each source files separately and then linking the resulting object files. You do not need to include any PHONY targets (like "all" or "clean"). Please use a variable for the compilation flags; the flags should include the flag for emitting all warnings (`Werror)` and for compiling in debug mode (`g`).

Question 3. [3 MARKS]

Processes

Part (a) [1 MARK]

Your friend has run the following code:

```
int main() {
    int my_val = 2;
    pid_t ret = fork();

    my_val = ret;

    printf("%d\n", my_val);
    return 0;
}
```

They claim that that they saw this output:

```
0
0
```

Please explain why you know they couldn't have seen that.

Part (b) [2 MARKS]

Write a piece of code that forks two children. The children should print "Child Done" and then exit. The parent should wait until both children have exited and then print "Parent Done" before exiting. You do not need to check for errors in this code.

```
int main() {
    pid_t ret;
```

[Use the space below for the processes question.]

Question 4. [7 MARKS]

Pointers Each of the subquestions below asks for a piece of code that uses the following `MarksNode` struct. The struct is intended to be used to build a linked list of mark components.

```
struct MarksNode {
    char *name;               // The name of the component, if specified.
    int mark;                 // The mark obtained.

    struct MarksNode *next;    // A link to the next item in the list.
};
```

Part (a) [1 MARK]

Fill in the body of the function below that frees the `MarksNode` that is passed in as an argument. Assume that both the node itself and the name inside were dynamically allocated.

```
void free_node(struct MarksNode *p) {
```

Part (b) [2 MARKS]

Assume that the pointer `list` points to the head of a linked list of `MarksNode`s. (It should continue to point to the head after your code is finished.) Remove all elements in the list with a mark below 10. You may use the function from the previous question to free the nodes you remove.

In the next two questions, you will write a function that takes two arguments – the location where the front of a list should be stored and an open file pointer – and builds a linked list from the contents of the file. The open file is a text file like the one below:

```
Midterm 14
Assignment1 13
Assignment2 25
```

Your function must work on files with more or fewer lines, but you may assume that any file will have a similar line format and, in particular, that none of the names in a file contain whitespace. Your function should return a 0 on success and a 1 if any errors occur.

Part (c) [1 MARK]

First, write just the declaration (the signature) of the function.

Part (d) [3 MARKS]

Next, assume that you have already written code to read one line of the file and to create the first node in the list. (Remember: the first node in a list is a special case. This keeps you from writing that special case.) The pointer `curr_node` points to this first node. Write the rest of the body of your function: it should read the remainder of the file, creating one node per line and adding it to the list. When it is done, it should return the appropriate value. *You should perform error checking on any library or system calls used. Your code may return immediately once an error is detected.*

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

C function prototypes:

```
pid_t fork(void);
int fclose(FILE *stream)
FILE *fopen(const char *file, const char *mode)
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
void free(void *ptr)
int fscanf(FILE *restrict stream, const char *restrict format, ...);
int fseek(FILE *stream, long offset, int whence)
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
int lstat(const char *restrict path, struct stat *restrict buf);
void *malloc(size_t size);
void perror(const char *s)
int scanf(const char *restrict format, ...);
size_t strlen(const char *s)
int strncmp(const char *s1, const char *s2, size_t n)
char *strncpy(char *dest, const char *src, size_t n)
char *strstr(const char *haystack, const char *needle)
pid_t wait(int *stat_loc);
```

Excerpt from the fseek man page:

If whence is set to SEEK_SET, SEEK_CUR, or SEEK_END, the offset is relative to the start of the file, the current position indicator, or end-of-file, respectively.

Excerpt from the wait man page:

WIFEXITED(status)
: True if the process terminated normally by a call to _exit(2) or exit(3).

WEXITSTATUS(status)
: If WIFEXITED(status) is true, evaluates to the low-order 8 bits of the argument passed to _exit(2) or exit(3) by the child.

Print your name in this box.

END OF EXAMINATION