CSC 209H1 S 2015 Midterm Test
Duration — 50 minutes
Aids allowed: none

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

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

Lecture Section: L0201 **Instructor: Simion**

---

*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 5 questions on 8 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. They may also get you part marks if you can't figure out how to write the code. Answers that contain both correct and incorrect or irrelevant statements will not get full marks.
If you use any space for rough work, indicate clearly what you want marked.

# 1:	_____/ 5
# 2:	_____/ 3
# 3:	_____/ 3
# 4:	_____/ 3
# 5:	_____/13
TOTAL:	_____/27

LEC 0201

Question 1. [5 MARKS]

Part (a) [1 MARK]

If your current working directory is `/home/user` give 2 different ways to run a program called `markone` that is stored in `/home/user/bin`.

Part (b) [3 MARKS]

A user types the following at the command line:

```
ant bug cat < dog > eel
```

The program or programs to be executed are: _____

The arguments to the program or programs are: _____

For each of the processes created, standard input comes from: _____

For each of the processes created, standard output goes to: _____

For each of the processes created, standard error goes to: _____

Part (c) [1 MARK]

Write a line of code that divides integer variable `a` by 2 without using the division operator (/).

Question 2. [3 MARKS]

Circle the correct answer, and briefly explain it.

TRUE FALSE It is possible that a parent process exits before its child process finishes. Explain:

TRUE FALSE In a parent process a wait call always blocks. Explain:

TRUE FALSE A process running in the background can't be killed. Explain:

Question 3. [3 MARKS]

Suppose we write a Makefile for assignment 1 with the following rules:

```
addecho : addecho.c
    gcc -Wall -g -o addecho addecho.c
remvocals : remvocals.c
    gcc -Wall -g -o remvocals remvocals.c
```

Part (a) [1 MARK] Write a rule so that when we run `make` both programs are recompiled if necessary.

Part (b) [2 MARKS] Write a rule with the target `test` that will ensure that `remvocals` is compiled if it is out of date, and will run `remvocals` with the arguments `simple.wav testout.wav`

Question 4. [3 MARKS]

Consider the following program.

```
int main() {

    int r = fork();

    if(r > 0) {
        int status;
        fprintf(stderr, "A");

        if(wait(&status) != -1) {
            if(WIFEXITED(status)) {
                fprintf(stderr, "%d", WEXITSTATUS(status));
            }
        }
        fprintf(stderr, "B");

    } else {
        fprintf(stderr, "C");
    }
    fprintf(stderr, "D");
    return 4;
}
```

Check all of the boxes that are valid output for the above program.

- ☐ A4BCDD
- ☐ CAD4BD
- ☐ BA4CDD
- ☐ AC4BDD
- ☐ ACD4BD
- ☐ ADC4BD

Question 5. [13 MARKS]

Given the following struct definition:

```
struct entry {
    char word[16];
    char def[64];
};
```

Part (a) [6 MARKS] Complete the code below so that it is correct, and the output is:

```
quokka : small macropod
macropod : marsupial family that includes kangaroos

// Store newword and newdef in the fields of e

void set_entry(_______________________________ e, char *newword, char *newdef) {









}

void print_entry(_________________________________ e) {


    printf("%s : %s\n", ____________________________, _____________________);
}

int main() {

    struct entry *dict = ____________________________________________________;


    set_entry(_______________________________, "quokka", "small macropod");


    set_entry(_______________________________, "macropod",
                                 "marsupial family that includes kangaroos");


    print_entry(____________________);


    print_entry(____________________);

}
```

Part (b) [1 MARK] How many bytes are required to store the following two variables.

`struct entry *dict:` _____ `struct entry e:` _____

Part (c) [6 MARKS]

Give the declaration for the variable to make each of the following statements correct, or if there is an error in the statements write "ERROR".

```
struct entry *dict;
struct entry words[5];
```

	Declaration of x
`x = &dict;`	
`x = dict->word`	
`x = words[1]`	
`x = &words[1]`	
`x = words[1].word[2]`	
`x = dict[0].word`	

C function prototypes and structs:

```
int execlp(const char *file, char *argv0, ..., (char *)0)
int execvp(const char *file, char *argv[])
int fclose(FILE *stream)
char *fgets(char *s, int n, FILE *stream)
pid_t fork(void)
FILE *fopen(const char *file, const char *mode)
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
int fseek(FILE *stream, long offset, int whence);
      /* SEEK_SET, SEEK_CUR, or SEEK_END*/
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
char *index(const char *s, int c)
void perror(const char *s);
unsigned int sleep(unsigned int seconds)
int sprintf(char *s, const char *format, ...)
int stat(const char *file_name, struct stat *buf)
char *strchr(const char *s, int c)
size_t strlen(const char *s)
char *strncat(char *dest, const char *src, size_t n)
int strncmp(const char *s1, const char *s2, size_t n)
char *strncpy(char *dest, const char *src, size_t n)
char *strrchr(const char *s, int c)
int wait(int *status)
int waitpid(int pid, int *stat, int options) /* options = 0 or WNOHANG*/
ssize_t write(int d, const void *buf, size_t nbytes);

WIFEXITED(status)      WEXITSTATUS(status)
WIFSIGNALED(status)    WTERMSIG(status)
WIFSTOPPED(status)     WSTOPSIG(status)
```

Useful structs

```
struct stat {
    dev_t st_dev; /* ID of device containing file */
    ino_t st_ino; /* inode number */
    mode_t st_mode; /* protection */
    nlink_t st_nlink; /* number of hard links */
    uid_t st_uid; /* user ID of owner */
    gid_t st_gid; /* group ID of owner */
    dev_t st_rdev; /* device ID (if special file) */
    off_t st_size; /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for file system I/O */
    blkcnt_t st_blocks; /* number of 512B blocks allocated */
    time_t st_atime; /* time of last access */
    time_t st_mtime; /* time of last modification */
    time_t st_ctime; /* time of last status change */
};
```

Print your name in this box.

End of Examination