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

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

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

Lecture Section: L5101 **Instructor: Reid**

---

*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: _____/ 6
2: _____/ 3
3: _____/ 2
4: _____/13
5: _____/ 5

TOTAL: _____/29

LEC 5101

Question 1. [6 MARKS]

Part (a) [2 MARKS]

Which of the following are **valid** ways to run the program `gnuplot` that is stored in `/usr/local/bin`.
`PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/u/reid/bin:.:`
`PWD=/usr/csc209/bin`

- ☐ `gnuplot`
- ☐ `./gnuplot`
- ☐ `/usr/local/bin/gnuplot`
- ☐ `../gnuplot`
- ☐ ` /gnuplot`
- ☐ `usr/local/bin/gnuplot`
- ☐ `../../local/bin/gnuplot`

Part (b) [1 MARK]

Write one line that you would type into the bash shell to set the permissions for the file `prog` to remove read write and execute permissions for everyone who is not the owner of the file or in the group.

Part (c) [2 MARKS]

Write one line that you would type into the bash shell to run the program `markall` found in the current directory so that it reads standard input from the file `classlist` and send the output to the program `grep` with the argument `TOTAL`. The output from `grep` should be saved in a file `results`.

Part (d) [1 MARK]

Write a line of code that sets all but the 5th bit in the integer variable `a`. The 5th bit remains unchanged.

Question 2. [3 MARKS]

Circle the correct answer, and briefly explain it.

TRUE FALSE File descriptors are inherited across both fork and exec calls
Explain:

TRUE FALSE A child process whose parent calls `wait` cannot become a zombie.
Explain:

TRUE FALSE If the PATH variable does not have the current working directory in it, then when you want to execute a program in the current working directory, you need to add `./` to the front.
Explain:

Question 3. [2 MARKS]

Consider the following makefile:

```
dotrans : dotrans.o list.o list.h
  gcc -Wall -g -o dotrans dotrans.o list.o

%.o : %.c list.h
  gcc -Wall -g -c $<
```

The contents of the current working directory are listed below:

```
-rw-r--r--  1 reid   268 11 Jan 23:07 Makefile
-rw-r--r--  1 reid  2673 11 Jan 23:07 dotrans.c
-rw-r--r--  1 reid  2710 11 Jan 23:07 list.c
-rw-r--r--  1 reid   270 11 Jan 23:07 list.h
-rw-r--r--  1 reid  5840  4 Mar 09:30 list.o
```

Part (a) [1 MARK] If we run `make`, which new files are added?

Part (b) [1 MARK] If we modify `list.h` and then run `make` again, which files are modified?

Question 4. [13 MARKS]

Part (a) [8 MARKS] Complete the code below so that it is operates correctly assumming the code that was omitted correctly initializes the data structures.

```
struct syn {
    char word[16];
    char *synonyms[NUMSYM];
};

/* initialize a struct syn with a word, and set all the elements of synonyms
 * NULL */

void init_syn(_________________________s, char *new_word){









}
/* Returns 1 if syn is in the list of synonyms for the word in entry "s",
 * and 0 otherwise
 */
int is_syn(___________________________________s, char *syn) {









}

// CONTINUED on next page
```

```
int main() {
    struct syn *thesaurus = malloc(SIZE * sizeof(struct syn));
    // call init_syn using the 0th element of thesaurus


    init_syn(____________________________, "excel");
    // omitted code that adds words and synonyms

    if(is_syn(thesaurus[0], "shine")) {
        printf("yes");
    }
}
```

Part (b) [5 MARKS]

Give the declaration for the variable `x` for each of the statements below, or if there is an error in the statements write "ERROR". Assume that the variables `thesaurus` and `words` are appropriately initialized and memory has been allocated.

```
struct syn *thesaurus;
struct syn words[5];
```

	Declaration for x
`x = thesaurus[1].synonyms[2];`	
`x = thesaurus[0].word[0];`	
`x = &thesaurus;`	
`x = *(thesaurus->word);`	
`x = words[0]->synonyms[1];`	

Question 5. [5 MARKS]

Consider the following program:

```
int main() {

    int r = fork();

    if(r == 0) {
        fprintf(stderr, "A");
        int q = fork();

        if(q == 0) {
            fprintf(stderr, "B");
            exit(1);
        } else {
            fprintf(stderr, "C");
            exit(1);
        }

    } else {
        int status;
        fprintf(stderr, "D");
        wait(&status);
        fprintf(stderr, "E");
    }
    return 0;
}
```

Part (a) [1 MARK] How many processes including the original one are created? _____

Part (b) [3 MARKS] Check the boxes beside the lines that are valid output for the program.

- ☐ ABCDE
- ☐ ACBDE
- ☐ ADEBC
- ☐ ADCEB
- ☐ ADBEC
- ☐ DACEB

Part (c) [1 MARK]

Is it possible for a child process in this program to ever become an orphan? Briefly explain your answer.

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