

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 7 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

---

## SOLUTIONS

## 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.

```
chmod o-rwx prog
```

### 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`.

```
markall < classlist | grep TOTAL > 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.

```
a = a | ~(1 << 5);
```

**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?

dotrans dotrans.o

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

dotrans dotrans.o list.o

**Question 4.** [13 MARKS]

**Part (a)** [8 MARKS] Complete the code below so that it operates correctly assuming 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");
    }
}

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(struct syn *s, char *new_word){
    strncpy(s->word, new_word, 16);
    int i;
    for(i = 0; i < NUMSYM; i++){
        s->synonyms[i] = NULL;
    }
}

/* Returns 1 if syn is in the list of synonyms for the word in entry "s",
 * and 0 otherwise
 */
int is_syn(struct syn s, char *syn) {
    int i;
    for(i = 0; i < NUMSYM; i++) {
        if(s.synonyms[i] != NULL &&
            (strcmp(s.synonyms[i], syn) == 0)) {
            return 1;
        }
    }
    return 0;
}

int main() {
    struct syn *thesaurus = malloc(SIZE * sizeof(struct syn));
    // call init_syn using the 0th element of thesaurus
    init_syn(&thesaurus[0], "excel");
}

```

```

// omitted code

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
<code>x = thesaurus[1].synonyms[2];</code>	<code>char *</code>
<code>x = thesaurus[0].word[0];</code>	<code>char x;</code>
<code>x = &amp;thesaurus;</code>	<code>struct syn **x</code>
<code>x = *(thesaurus-&gt;word);</code>	<code>char x;</code>
<code>x = words[0]-&gt;synonyms[1];</code>	ERROR (needs a . rather than ->)

**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? 3

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

☒ ABCDE

☒ ACBDE

☐ ADEBC

☒ ABDCE

☐ 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.

*Yes. The child of the child could run slowly and the original process and the first child could terminate first.*