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

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

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

Lecture Section: L0101 **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: _____/ 5
5: _____/13

TOTAL: _____/29

SOLUTIONS

LEC 0101

Question 1. [6 MARKS]

Part (a) [1 MARK]

Write one line that you can type at the bash shell prompt to change to the parent directory of your home directory.

SOLUTION:

```
cd \~/..
cd $HOME/..
```

Part (b) [1 MARK]

Run the program `markall` found in the current directory so that it reads standard input from the file `classlist`, saves the standard output in a file `results`, and leaves standard error going to the console.

SOLUTION:

```
markall < classlist > results
```

Part (c) [1 MARK]

Write a snippet of C code that prints "yes" if the bit at index 3 in the variable `perms` is set, and prints "no" otherwise.

```
unsigned short perms;
```

SOLUTION:

```
if(perms & 0b1000) { //or 8
    printf("yes");
} else {
    printf("no");
}
```

Part (d) [3 MARKS]

A user types the following on the command line:

```
red green blue | grey >& white
```

Fill in the table below for each of the processes in the command line. Note that there may be more columns than necessary.

Program name	red	grey	
Argument(s)	green blue		
Standard input comes from	console	pipe (or red)	
Standard output goes to	pipe (or grey)	white	
Standard error goes to	console	white	

Question 2. [3 MARKS]

Circle the correct answer, and briefly explain it.

TRUE **FALSE** A child process has the same address space as its parent.

Explain:
The child and parent processes do not share memory. They each have their own address space.

TRUE FALSE In a parent process, the wait() call blocks until at least one child signals the parent with its exit status.

Explain:
This is true unless the process never created any child processes.

TRUE FALSE Global variables should never be declared in header files.

Explain:
A variable declaration means that memory is allocated to store a value for the variable. No statements that allocate memory should be put in a header file, since the header file may be included multiple times.

Question 3. [2 MARKS]

Consider the following makefile:

```
simpletest : simpletest.o smalloc.o testhelpers.o
  gcc -Wall -g -o simpletest simpletest.o smalloc.o testhelpers.o

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

The current working directory contains the following files

```
Makefile    simpletest.c    smalloc.c   smalloc.h   testhelpers.c
```

Part (a) [1 MARK] If we run `make`, which new files are added? SOLUTION:
`simpletest.o, smalloc.o, testhelpers.o, simpletest`

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

SOLUTION:
`smalloc.o simpletest`

Question 4. [5 MARKS]

Consider the following program:

```
int main() {

    if(fork() == 0) {
        if(fork() == 0) {
            fprintf(stderr, "B");
            exit(3);
        } else {
            fprintf(stderr, "B");
            exit(2);
        }

    } else {
        int status;
        fprintf(stderr, "C");
        if(wait(&status) != -1) {
            if(WIFEXITED(status)) {
                fprintf(stderr, "%d", WEXITSTATUS(status));
            }
        }
        fprintf(stderr, "D");
    }
    fprintf(stderr, "E");
    return 0;
}
```

Part (a) [1 MARK] How many processes including the original one are created? 3

Part (b) [3 MARKS]

Is more than one output ordering possible? If so, write all of the possible orders. If not, write the output, and explain why it must be the only order.

CBB2DE BCB2DE BBC2DE

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.

Question 5. [13 MARKS]

Part (a) [8 MARKS] Complete the code below so that it is correct, and the `contacts` contains:

```
{"pizza pizza", "416-967-1111"}, {"ghostbusters", "555-3268"}
```

```
struct contact {
    char *name;
    char phone[16];
};
void set_contact(_________________________ c, char *new_name, char *new_phone) {









}
/* Add "code" to the beginning of the number in "num".  Assume that the caller
 * has already ensured that there is enough space in "num" to add "code". */
void add_area_code(char *code, char *num) {









}
int main() {
    struct contact *contacts = malloc(_________________________________________);

    // Set the the zero'th element of contacts


    set_contact(_____________________________, "pizza pizza", "416-967-1111");

    // Set the next element of contacts


    set_contact(_____________________________, "ghostbusters", "555-3268");

    // add area code to the phone number for "ghostbusters"


    add_area_code("416-", ____________________________);
```

```
}


struct contact {
    char *name;
    char phone[16];
};

void set_contact(struct contact *e, char *new_name, char *new_phone) {
    // malloc wasn't strictly necessary.  We could just assign the pointer
    // without a copy.
    e->name = malloc(strlen(new_name) + 1);
    strncpy(e->name, new_name, 16);
    strncpy(e->phone, new_phone, 16);
}
/* add the code to the beginning of the number in num.  Assume that
* the caller has already ensured that there is enough space in num
* to add code.
** This is a poor but workable solution*/
void add_area_code(char *code, char *num) {
    char tmp[256];
    strncpy(tmp, code, 256);
    strncat(tmp, num, 256);
    strncpy(num, tmp, strlen(num) + strlen(code) + 1);
}
void print_contact(struct contact e) {
    printf("%s : %s\n", e.name, e.phone);
}

int main() {
    struct contact *contacts = malloc(10 * sizeof(struct contact));

    set_contact(&contacts[0], "pizza pizza", "416-967-1111");
    set_contact(&contacts[1], "ghostbusters", "555-3268");

    print_contact(contacts[0]);
    print_contact(contacts[1]);
}
```

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 `contact` and `clist` are appropriately initialized and memory has been allocated.

```
    struct contact *contact;
    struct contact clist[5];
```

	Declaration for x
`x = &contact;`	`struct contact **x;`
`x = contact[0].name[0]`	`char x;`
`x = &(contact[1].phone)`	`char (*x)[])` (we accepted "char **" or ERROR since this is too subtle)
`x = &(contact[1].phone[3])`	`char *x`
`x = *contact`	`struct contact x`

End of Solutions