CSC 209H1 S 2018 Midterm Test
Duration — 50 minutes
Aids allowed: none

UTORID: |_|_|_|_|_|_|_|_|_|_|

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

Section: L5101 (W6-8)
Instructor: Craig

---

*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 8 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
Comments are not required.
No error checking is required.
You do not need to provide the include statements for your programs.
If you use any space for rough work, indicate clearly what you want marked.

1: _____/ 5
2: _____/ 6
3: _____/ 8
4: _____/ 6

TOTAL: _____/25

Question 1. [5 MARKS]

Assume you have a terminal open, and the current working directory contains a C program file called `wizard.c` and a text file called `potion.txt`.

Part (a) [1 MARK]

The following command is used to compile `wizard.c` into an executable:

```
gcc -Wall -o spell -g -std=99 wizard.c
```

Write a shell command to execute the executable file produced by the statement above with command line argument `confundo` and write the output to `potter` instead of to standard out.

Part (b) [1 MARK]

Give a one-line shell command to change the permissions on `spell` so that is readable by everyone, executable by the users who owns the file and members of the group and writeable by no-one.

Part (c) [3 MARKS]

You have a binary file containing only 100 consecutive integers such that the first integer is at the beginning of the file (starting in byte 0.) Add to the code fragment below, to efficiently read the value for the 42nd integer into the variable `i`;

```
int i;
FILE *fp = fopen("someBinaryFile", "r");
```

Question 2. [6 MARKS]

For each of the code fragments below, there is missing code. At the very least, the line (or lines) that declare and possibly initialize the variable `x` are missing. If the code will not compile no matter what you put for the missing code, check `COMPILE ERROR` and explain why. If the code will compile, but is not guaranteed to run without an error, check `RUN-TIME ERROR` and explain why. Otherwise, check `NO ERROR` and give the correct declaration for x. You don't need to show any other missing code. The first one is done for you.

Code Fragment	ERROR	Declaration for `x` or explanation

`int y;`
`// missing code`
`x = y;`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`int x;`

`int **p = malloc(3 * sizeof(int *));`
`// missing code`
`x = *(p + 1);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`struct person {`
`  char *name;`
`};`
`// missing code`
`x->name = "???";`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`int **p = malloc(sizeof(int**));`
`// missing code`
`*p = &x;`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`char *s = malloc(6 * sizeof(char));`
`strcpy(s, "hello");`
`// missing code`
`x = **s;`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`char *course = "csc207";`
`// missing code`
`x = course;`
`x[5] = '9';`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`void fun(XXX type_hidden) {`
`  // missing code`
`  x[1] = type_hidden[2];`
`}`
`int main() {`
`  int array[3] = {1, 7, -4};`
`  fun(array);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

Question 3. [8 MARKS]

Complete this program that uses a string in the format `"MonthName DayNumber"` (e.g. `December 25`), creates an `event` struct with the fields set based on the string, and prints the information in a different format.

```
struct event {
    char *month;
    int day;
};

/* Set the month and day of an event struct given a string in the format
 * "MonthName DayNumber". For example, if line is "December 25",
 * the event's month should be set to "December", and its day should
 * be set to 25.
 * Do not change the variable line.
 * Do not allocate more memory than you need.
 */

void set_struct(_____________________________________________, char *line) {









}
```

```
/* Return the integer representation of month. For example, get_month_as_num("January")
 * should return 1.  If month is not a valid month, (not in months below), return 0.
 */
int get_month_as_num(char *month) {
    char *months[] = {"January", "February", "March", "April", "May", "June",
        "July", "August", "September", "October", "November", "December"};









}
int main() {
    char line[MAX];
    // Hidden code that sets line to a string of the correct format.
    // Create a new event and call set_struct to set the fields.

    struct event *new_event = malloc(_____________________________________________________);

    set_struct(_____________________________________________________, line);

    // Print the event information in the following format:
    //     Event Date: December (12) 25
    // You should use get_month_as_num().


    printf("Event Date: %s (%d) %d", ____________________________________________ ,


    ______________________________________, ________________________________________________);

    // Free all dynamically-allocated memory









    return 0;
}
```

Question 4. [6 MARKS]

The function `intertwine()` takes two strings as arguments, and returns a dynamically allocated string containing a mix of both arguments. Characters in the returned string should be added in the following order: first character of the first argument, first character of the second argument, second character of the first argument, second character of the second argument, etc. until all characters have been added.

Some examples of input and output can be seen below:

```
intertwine("hello", "world") -> "hweolrllod"
intertwine("go on" "odmring") -> "good morning"
intertwine("CSC209", "") -> "CSC209"
```

You may assume that both `s1` and `s2` are strings. That is, they are null-terminated.

Write the function `intertwine()` according to the above description.

```
char *intertwine(char *s1, char *s2) {




}
```

C function prototypes:

```
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)
void *malloc(size_t size)
void perror(const char *s)
int scanf(const char *restrict 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)
char *strstr(const char *haystack, const char *needle)
```

Excerpt from strcpy/strncpy man page:

```
The stpcpy() and strcpy() functions copy the string src to dst (including
the terminating ‘\0’ character).

The stpncpy() and strncpy() functions copy at most n characters from src
into dst.  If src is less than n characters long, the remainder of dst is
filled with ‘\0’ characters.  Otherwise, dst is not terminated.
```

Excerpt from strstr man page:

```
strstr(const char *haystack, const char *needle)
The strstr() function finds the first occurrence of the substring needle
in the string haystack. It returns a pointer to the beginning of the
substring, or NULL if the substring is not found.
```

Excerpt from strchr man page:

```
The strchr() function locates the first occurrence of c (converted to a char) in the
string pointed to by s.  The terminating null character is considered to be part of
the string; therefore if c is ‘\0’, the functions locate the terminating ‘\0’.

The strrchr() function is identical to strchr(), except it locates the last occurrence of c.
```

Print your name in this box.

End of Examination