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

UTORid: _____

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

Section: L0101 (MW10)
Instructor: Campbell

---

*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 6 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 for system calls.
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: _____/ 5
4: _____/ 9

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 magic -g -std=gnu99 wizard.c
```

Write a shell command to execute the executable file produced by the statement above with input redirected so the program reads from `potion.txt` rather than from standard input.

Part (b) [2 MARKS]

Now, a header file named `wizard.h` is added to the current working directory and `wizard.c` is revised to include that header file.

Write a Makefile with one rule so that when you type `make wizard`, the executable program will be created (or re-created) only if one or both of the files `wizard.c` or `wizard.h` have been modified. Note that the program should be compiled with the -Wall and -g flags.

Part (c) [1 MARK]

The following command is used to set the permissions of `potion.txt`:

```
chmod 731 potion.txt
```

Check the boxes to indicate the file permissions of `potion.txt` after the command above has been executed.

user:	☐ read	☐ write	☐ execute
group:	☐ read	☐ write	☐ execute
other:	☐ read	☐ write	☐ execute

Part (d) [1 MARK]

In the box, print the number of bytes that will be written to the file by this code fragment.

```
int i = 82;
fprintf(fp, "%d\n", i);
```

Question 2. [6 MARKS]

For each code fragment below, if the code will not compile or will generate a warning when compiled with the `-Wall` flag, 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 show what is printed. The first one is done for you.

Code Fragment	ERROR	Output or explanation for error

`int y = 2;`
`int x = y;`
`printf("%d %d", x, y);`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		2 2

`char *name = "Me";`
`printf("%s\n", name);`
`free(name);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`char *animal = "cat";`
`animal[0] = 'r';`
`printf("%s", animal);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`char s[] = "RaCecAr";`
`char *p = s;`
`int q = 6;`
`while (p <= &s[q]) {`
`  if (*p != s[q]) {`
`  *p = s[q];`
`  }`
`  p++;`
`  q--;`
`}`
`printf("%s", s);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`char *w = malloc(3 * sizeof(char));`
`strcpy(w, "AB");`
`char *m[2];`
`m[0] = w;`
`m + 1 = w + 1;`
`printf("%s, %s", *m, *(m + 1));`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`char food[6];`
`char *fruit = "banana";`
`food = fruit;`
`printf("%s", food);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

`char s[10] = "computer";`
`s[6] = '\0';`
`printf("%s", s);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR	

Question 3. [5 MARKS]

Write the function `count_occurrences` according to its description below. Complete the main function.

Notice that it counts overlapping occurences. For example, given `"ababa"` and `"aba"` for the first two arguments, the function should update the memory pointed at by `result` to `2`.

You may not use `strstr()`, nor any other string library function that would find a substring for you.

```
/* Find the number of times substr appears in str (including overlapping occurrences) and
 * set the memory pointed at by result to that number.  If str and substr are identical,
 * set the memory pointed at by result to 1. */
void count_occurrences(char *str, char *substr, int *result) {









}
int main() {

    int res;
    // Call count_occurrences on "ababa" to find "aba" and set res.



    printf("aba occurred %d times in ababa\n", res);
    return 0;
}
```

Question 4. [9 MARKS]

Complete the following program according to the instructions in the comments. Assume that all system calls succeed and that the arguments are of the specified format. Only allocate the space you need.

```
struct flight {
    char *code;
    int seats_available;
};
/* Return a pointer to a new struct flight with its code and seats_available
 * initialized to the data given in arg.  arg will be a flight code (e.g., AC1123)
 * followed by a colon (:) and the number of seats on that flight (e.g., 100).
 * Once a flight is created, its code must refer to a dynamically-allocated string. */
struct flight *create_flight(char *arg) {









}
/* If there are enough seats available on f, book num_seats; Otherwise do nothing. */
void book_seat(struct flight* f, int num_seats) {




}
int main(int argc, char **argv) {
    // argv[1] represents a flight in the format code:seats (e.g., AC1123:100, WSG324:24)
    struct flight *my_flight = create_flight(argv[1]);
    book_seat(my_flight, 2);
    // Free memory



    return 0;
}
```

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