CSC 209H1 S 2017 Final Test
Duration — 50 minutes
Aids allowed: none

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

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

Instructor: Reid
Section: L5101 (6:10-7:00pm)

---

*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.
No error checking is required unless you are specifically requested to do it for an individual question.
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: _____/ 8
2: _____/ 2
3: _____/ 4
4: _____/ 6
5: _____/ 6

TOTAL: _____/26

Question 1. [8 MARKS]

The following program reads star data from a text file in the following format. Each line contains a string (with no spaces in it) representing the name of a star followed by a space and the magnitude of the star as a floating point number. In writing the code below, you may assume that the file has the correct format.

Example:

```
Polaris 1.97
Vega 0.03
Aldebaran 0.87
```

Part (a) [5 MARKS]

Complete the function `read_stars` that will read star data from the open text file pointer passed in as an argument. It will create and build a linked list of `struct star`'s from this data.

- New elements may be inserted at the head of the list.
- If the file is empty, return NULL
- Assume the length of a star name is no more than `MAXSIZE - 1`.
- Values stored in the linked list should occupy no more space than necesssary.

```
struct star {
    char *name;
    double magnitude;
    struct star *next;
};

struct star *read_stars(File *fp)  {
```

Part (b) [3 MARKS]

Complete the function below that calls free on memory allocated for the linked list of stars, `head`. Remember not to leave dangling pointers.

```
void free_stars(struct star *head) {
```

Question 2. [2 MARKS]

I spent a long time working on my `print_ftree` program on cdf and it works perfectly, but when I compile and run it on another Linux machine, once in a while it gives me a segmentation fault.

Part (a) [1 MARK] Is there a bug in my code? Explain your answer.

Part (b) [1 MARK] When does the segmentation fault message appear? Check the appropriate answer or answers.

- ☐ Compile time
- ☐ Run time
- ☐ When runnning `make print_ftree`

Question 3. [4 MARKS]

Consider the following Makefile from Assignment 1

```
FLAGS = -Wall -std=c99

compute_hash: compute_hash.o hash_functions.o
	gcc ${FLAGS} -o $@ $^

%.o : %.c
	gcc ${FLAGS} -c $<

clean :
	rm *.o compute_hash
```

The contents of the current working directory are:

```
Makefile    compute_hash.c  hash_functions.c
```

In the questions below, identify the files that are created, modified or deleted when each command is run in sequence.

Part (a) [1 MARK] `make compute_hash.o`

Created	Modified	Deleted
`compute_hash.o`		

Part (b) [1 MARK] `make`

Created	Modified	Deleted
`hash_functions.o` `compute_hash`		

Part (c) [1 MARK]

First `hash_functions.c` is modified and then `make compute_hash` is run.

Created	Modified	Deleted

	`(hash_functions.c)`
	`hash_functions.o`
	`compute_hash`	

Part (d) [1 MARK]

I do some more work on the program and add a new file called `helper.c` that contains functions called by code in `compute_hash.c`. (Assume I updated the source files correctly.) Modify the Makefile above so that the new file will be compiled correctly with the program.

Question 4. [6 MARKS]

```
struct team {
    char name[16];
    char *players[4];
};
```

```
struct team *ont;
struct team scotties[10];
```

Part (a) [3 MARKS]

Give the declaration for the variable x in each of the statements below, or if there is an error in the statments, describe the error in a few words. Assume that `ont` and `scotties` (above) have been appropriately initialized with all necessary memory allocated.

Statement	Declaration of x or error
`x = ont->name;`	
`x = ont->name[0];`	
`x = &ont;`	
`x = scotties[2];`	
`x = scotties.players[1];`	
`x = &scotties[2].players[1];`	

Part (b) [3 MARKS]

For each of the following independent snippet of code identify the problem with the code, or write "no problem" if there are no potential problems with the snippet.

Statements	Problem
`ont->name = malloc(10);` `strncpy(ont->name, "Homan", strlen("Homan"));`	
`ont->name = malloc(10);` `ont->name = "Homan";`	
`char name[10] = "Homan";` `ont->name = name;`	

Question 5. [6 MARKS]

Complete the function below, so that the following examples will work correctly and the minimum amount of memory is used. The only string library functions you may use are `strlen` and `strncpy`.

```
char str[] = "first,,third,fourth";
printf("0 %s\n", getfield(str, 0));
printf("1 %s\n", getfield(str, 1));
printf("2 %s\n", getfield(str, 2));
printf("3 %s\n", getfield(str, 3));
```

Prints:

```
0 first,,third,fourth
1 ,third,fourth
2 third,fourth
3 fourth
```

```
/* Returns a new string containing the characters starting from
 * the nth comma in line until the end of the line.
 * Does not modify line.
 * Returns NULL if n is larger than the number of commas in line
 */

char *getfield(char *line, int n) {
```

C function prototypes:

```
int fclose(FILE *stream)
char *fgets(char *s, int n, FILE *stream)
FILE *fopen(const char *file, const char *mode)
int fprintf(FILE *stream, const char *format, ...)
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)
DIR *opendir(const char *name)
void perror(const char *s)
int printf(const char *format, ...)
struct dirent *readdir(DIR *dir)
int scanf(const char *restrict format, ...)
int lstat(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 fgets man page:

```
    fgets() reads in at most one less than size characters from stream  and
    stores  them  into  the buffer pointed to by s.  Reading stops after an
    EOF or a newline.  If a newline is read, it is stored into the  buffer.
    A  terminating  null  byte ('\0') is stored after the last character in
    the buffer.
```

Excerpt from scanf/fscanf man page:

```
RETURN VALUES
     scanf and fscanf return the number of input items assigned.  This can be
     fewer than provided for, or even zero, in the event of a matching fail-
     ure. The value EOF is returned if an input failure occurs before any
     conversion such as an end- of-file occurs.
```

Makefile variables: `$@` is the target, `$^` is the list of prerequisites `$<` is the first prerequisite.

Print your name in this box.

End of Examination