CSC 209H1 S 2012 Midterm Test
Duration — 50 minutes
Aids allowed: none

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

Last Name: ______________________ **First Name:** ______________________

Lecture Section: L5101 **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 6 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
Comments are not required except where indicated, 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.
If you use any space for rough work, indicate clearly what you want marked.

# 1:	_____/ 7
# 2:	_____/ 8
# 3:	_____/ 4
# 4:	_____/ 6
TOTAL:	_____/25

Question 1. [7 MARKS]

In each subquestion below, fill in the box with the declaration for an appropriate mystery function so that the following code would compile without error. The subquestions are independent.

Part (a) [4 MARKS]

```
double price;
int * position;
char ** products;
/* Assume that these variables are initialized properly here */

if (mystery1(&price, *position, products[3]) != 0) {
    fprintf(stderr, "Error with mystery 3\n");
}
```

Part (b) [3 MARKS]

```
int rating[3] = {4, 5, 10};
char ** items;
char name[30];
strncpy(name, mystery2(rating[2], &items), 29);
```

Question 2. [8 MARKS]

Part (a) [5 MARKS]

On the next page write a shell program `longer` that takes two filenames as command line arguments and prints to standard output the content of the file that is longer. By default, length is determined as the number of lines, but if the script is called with an optional -w argument (that must come before the filenames), it determines length based on the number of words instead.

For example, suppose that the two files are as shown here:

File1

```
File1 has only two lines
but lots of words
```

File2

```
File2
has
more
lines
```

If your function was called as `longer File1 File2`, it would print the contents of `File2`. If were called as `longer -w File1 File2`, it would print the contents of `File1`.

You may assume that the files corresponding to the arguments do exist and are readable.

Part (b) [3 marks]

In order to test your script for part a, you would like to create five files named `file1, file2, file3, file4,` and `file5`. They should have the number of lines corresponding to their name. For example, `file4` should have four lines. Complete the shell script below that would create these files in a way that only the `for` loop would need changing to create more than five of these files. It doesn't matter exactly what you put on the lines of the test files as long as they aren't empty. You may assume that the files do not exist, or are empty before the script runs.

```
for i in 1 2 3 4 5; do
  #create file i
```

Question 3. [4 MARKS]

Part (a) [1 MARK]

Two contstants `SIZE1` and `SIZE2` have valid, but unknown values. Suppose that `str2` is initialized to contain a string. Complete the line of code in the box below with a correct third argument to `strncpy` so that the result will be correct regardless of the values of `SIZE1`, `SIZE2` and `n`.

```
char str1[SIZE1];
char str2[SIZE2];

/* str2 is initialized to a valid string */
int n = strlen(str2);
```

```
strncpy(str1, str2,
```

Part (b) [2 MARKS]

Assuming the third argument to `strncpy` is correct, is it possible for `strlen(str1) > SIZE1`? If yes, give an example. If no, explain why not.

Part (c) [1 MARK]

In the box below this code fragment, print the output of this code.

```
int a[5] = {1,2,3,4,5};
int * p = a;
p++;
printf("%d",p[2]);
```

Question 4. [6 MARKS]

Part (a) [3 MARKS]

Consider the C code below that defines a struct to represent a gift card. Write a function `adjust_amount` which takes a card and a charge and attempts to reduce the balance of the card by the charge. If there is enough money on the card, the balance is reduced and the function returns 0. If the card doesn't have enough money to cover the charge, then the balance is reduced to 0 and the amount that was successfully paid by the giftcard is returned.

```
#define MAXNAME 24
struct giftcard{
    char name[MAXNAME];
    double balance;
};
```

Part (b) [3 MARKS]

Write C code that creates a struct to represent Karen Reid's gift card which has a balance of $3.98. Then call your function to spend $2.05 (enough for a starbuck's grande) on Karen's card.

C function prototypes and structs:

```
int fclose(FILE *stream)
char *fgets(char *s, int n, FILE *stream)
FILE *fopen(const char *file, const char *mode)
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
int fseek(FILE *stream, long offset, int whence)
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
char *index(const char *s, int c)
void perror(const char *s)
int sprintf(char *s, const char *format, ...)
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)
```

Shell comparison operators

Shell	Description
-d filename	Exists as a directory
-f filename	Exists as a regular file.
-r filename	Exists as a readable file
-w filename	Exists as a writable file.
-x filename	Exists as an executable file.
-z string	True if empty string
str1 = str2	True if str1 equals str2
str1 != str2	True if str1 not equal to str2
int1 -eq int2	True if int1 equals int2
-ne, -gt, -lt, -le	For numbers
!=, >, >=, <, <=	For strings
-a, -o	And, or.

```
expr match STRING REGEXP
expr ARG1 + ARG2
```

Useful Unix programs for shell programs: `cat, cut, wc, grep, sort, sort -n (for numerical sorting), head, tail`

Print your name in this box.