CSC 209H1 Y 2011 Midterm
Duration — 50 minutes
Aids allowed: none

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

Last Name: ____________________ **First Name:** ____________________

Lecture Section: 1 **Instructor: Daniel Zingaro**

*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.* If you use any space for rough work, indicate clearly what you want marked.

# 1:	_____/ 4
# 2:	_____/ 6
# 3:	_____/ 6
# 4:	_____/ 4
TOTAL:	_____/20

Question 1. [4 MARKS]

Part (a) [1 MARK]

What is the difference between `echo` and `echo -e`?

Part (b) [2 MARKS]

If you call `find` with the `-print0` option:

```
find . ... -print0
```

you get a list of files in the current directory, where each file is followed by a null character. And if you call `xargs` with the `-0` option, it uses the null character as the separator between arguments read from standard input.

Using the `print0` and `-0` options as described, write a Bourne shell pipeline that removes all files rooted in the current directory (including files in subdirectories) whose names end with `.tmp`.

Part (c) [1 MARK]

Consider the following sequence of commands from within a directory initially containing no files:

```
greywolf:~/csc209/tempdir$ echo hello>file1
greywolf:~/csc209/tempdir$ ln file1 file2
greywolf:~/csc209/tempdir$ ln -s file1 file3
greywolf:~/csc209/tempdir$ rm file1
```

Explain the result if I now run the command `cat file3`.

Question 2. [6 MARKS]

Write a Bourne shell program that produces one line of output, as follows, for each regular file in the current directory:

- If the filename contains no uppercase characters, print the filename.
- If the filename contains at least one uppercase character, print a `*` (asterisk) and then the filename.

For example, for a directory containing files `dan` and `joE` (capital E), the output would be:

```
dan
*joE
```

You must use wildcard expansion to loop through files (not `ls` or `find`). You can use a grep pattern of `[A-Z]` to search for an uppercase character.

Question 3. [6 marks]

Write the C function below that inserts a copy of `c2` after each instance of `c1` in `dest`. `c1` and `c2` are guaranteed to be unequal, and `dest` is guaranteed to have enough space to hold all new characters. Do not use any string functions besides `strlen`. The function returns a pointer to `dest`.

```
char *insert (char *dest, char c1, char c2) {
```

Question 4. [4 marks]

Part (a) [2 marks]

Explain the bug in the `makePerson` function below. (The bug is not in `main`; `main` just demonstrates a call of the function.)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct person {
  char *name;
  int age;
};

struct person *makePerson (void) {
  struct person p;
  p.name = malloc(10 * sizeof (char));
  strcpy (p.name, "Dan");
  p.age = 28;
  return &p;
}

int main(void) {
  struct person *p = makePerson();
  //...
  printf ("%s\n", p->name);
  free (p->name);
  free (p);
  return 0;
}
```

Part (b) [2 marks]

What is the output of the following program? If the program has an error, please describe the error rather than giving any output.

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  char **s = malloc(2 * sizeof (char *));
  char a[] = "Zelda 3";
  s[0] = a;
  s[1] = a + 3;
  *(s[1]) = 'l';
  printf ("%s\n", a);
  return 0;
}
```

C function prototypes

```
int printf(const char *format, ...)

char *strchr(const char *s, int c) //Search from left
char *strrchr(const char *s, int c) //Search from right
char *strstr(const char *s1, const char *s2) //Search for s2

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)
```

Shell `test` Operators

`-a`, `-o`	and, or
`-eq`, `-ne`	equality and inequality on ints
`-gt`, `-lt`, `-ge`, `-le`	greater-than, less-than, etc. on ints
`=`, `!=`	equality and inequality on strings
`-z`	Empty string?
`-x`	File is executable?
`-r`	File is readable?
`-w`	File is writable?
`-f`	File is a regular file?
`-d`	File is a directory?

Shell Commands

- `grep` returns 0 if no match was found, 1 if a match was found, 2 if error
- By default, `grep` prints lines that match (`-v` prints lines that do not match)
- `ln file1 file2` makes `file2` a hard link to the same file as `file1` (`-s` creates symbolic links)
- `tr SET1 SET2`
- `ps` (`-A` for all processes, `-f` for more columns)
- `sort`, `uniq`, `set`

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

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