CSC 209H1 S 2013 Midterm Test
Duration — 50 minutes
Aids allowed: none

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

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

Lecture Section: L0101 **Instructor: Reid**

---

*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 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: _____/ 9
2: _____/ 5
3: _____/ 4
4: _____/12

TOTAL: _____/30

Question 1. [9 MARKS]

In the box beside each set of `bash` statements, show the output when the statements are executed. If running these statements would result in an error, print ERROR and give a brief explanation.

Part (a) [2 MARKS]

```
first=Santa
last=Clause
name=first+last
echo name is $name
echo 'last is $last'
```

Part (b) [1 MARK]

```
# The current working directory
# contains the files a1 a2 b c

if [ "a1 a2" = "a*" ]
then
    echo match
else
    echo "a*"
fi
```

Part (c) [1 MARK]

```
# The current working directory
# contains the files a1 a2 b c

files="a*"
if [ "a1 a2" = $files ]
then
    echo match
else
    echo $files
fi
```

Part (d) [2 MARKS]

In the current directory, some of the files contain the string `FIX ME`. Write a shell command that counts the number of lines containing this phrase in the files ending in `.c`

Part (e) [3 MARKS]

Write a shell loop to create 3 files named `test_one`, `test_two`, `test_three` that each contain the word `one`, `two`, or `three` respectively. (Hardcoding the answer with no loop will receive 0 marks.)

Question 2. [5 MARKS]

Part (a) [2 MARKS]

In the box beside the code fragment below, provide the output. If the code results in an error before anything is printed, write ERROR and give a brief explanation.

```
char s[6] = "01234";
char * t = "XX";
strncpy(s+2, t, 3);
printf("%s\n", s);
```

Part (b) [1 MARK]

What is the type of `&t`?

Part (c) [2 MARKS]

Provide the output of the following program.

```
int double1(int x) {
    x = x + x;
    return x;
}

int double2(int *x) {
    *x = (*x) + (*x);
    return *x;
}


int main() {
    int i = 10;
    int j = 0;

    j = double1(i);
    printf("i = %d, j = %d\n", i, j);       //Output:_____________________________

    i = 10;
    j = double2(&i);
    printf("i = %d, j = %d\n", i, j);       //Output:_____________________________
    return 0;
}
```

Question 3. [4 MARKS]

Suppose the current working directory contains the following files, and that the `makefile` contains the rules in the box to the right:

```
a3          helpers.h   t2.txt
a3.c        makefile
helpers.c   t1.txt
```

In the table below are four commands that are run sequentially (one after the other) from the shell. Complete the table by filling in the line numbers of the actions that are executed in and by filling in the names of files that are changed or created.

```
1    all_tests: test1 test2

2    test1: a3
3           a3 t1.txt

4    test2: a3
5           a3 t2.txt

6    a3 : a3.o helpers.o helpers.h
7          gcc ${CFLAGS} -o a3 a3.o helpers.o

8    %.o : %.c helpers.h
9           gcc ${CFLAGS} -c $@
```

Command	Line numbers of executed actions	Names of Files Created or Changed
`make a3`		
`make test2`		
`rm -rf a3`	No actions	a3 is removed
`make all_tests`		

Question 4. [12 MARKS]

Part (a) [7 MARKS] Consider the C program below which produces the output:

```
Even positions are 0246 and odd positions are 1357
Even positions are ACE and odd positions are BD
```

Complete the function `parity_strings` as described in the function comment. Your program should generate this output without making any changes to `main`. Your function must not change parameter `s`.

```
/* Return a pointer to an array of two strings. The first is the characters  of s that
 * are at even indices and the second is the characters from s at odd indices */

char ** parity_strings(char * s) {
```

```
int main() {
    char s1[9] = "01234567";
    char ** r = parity_strings(s1);
    printf("Even positions are %s and odd positions are %s\n", r[0],r[1]);

    char * s2 = "ABCDE";
    r = parity_strings(s2);
    printf("Even positions are %s and odd positions are %s\n", r[0],r[1]);
}
```

Part (b) [1 MARK]

We explicitly stated that your function could not change parameter `s`. State when and why the program would fail if you didn't follow this instruction.

- Which call to `parity_strings` would fail? (pick one)

  ☐ Only the first call on `s1` would fail.
  ☐ Only the second call on `s2` would fail.
  ☐ Both calls would fail.

- Why would it fail?

Part (c) [1 MARK]

The program as written has a memory leak in the `main` function. Explain where this is?

Part (d) [3 MARKS]

Complete a function `free_all` that could be added to the program to fix the memory leak. Write the function here, but add the call(s) to the function into the main function on the previous page.

```
void free_all(                              ) {
```

C function prototypes:

```
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);
void free(void *)
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 *malloc(size_t size);
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)
```

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

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

Print your name in this box.

End of Examination