Midterm Test

CSC 209H1S / LEC 0101/0201/L0301 — Campbell

February 27 2020, 2:10pm — Duration: **50 minutes**

Aids Allowed: **None**

First (Given) Name(s):

Last (Family) Name(s):

10-Digit Student Number:

UTORid (e.g., `pitfra12`):

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, write your name, student number, and UTORid above
(please do this now!) and read the instructions below *carefully*.

- This term test consists of 4 questions on 8 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete.*

- Answer each question directly on the test paper, in the space provided, and use a "blank" page for rough work. If you need more space for one of your solutions, use one of the "blank" pages and *indicate clearly the part of your work that should be marked.*

- Comments are not required.

- No error checking is required.

- You do not need to provide the include statements for your programs.

- Do not remove any pages from the exam booklet.

Good Luck!

Question 1. [4 MARKS]

Assume you have a terminal open, and the current working directory contains a C program file called guess.c and two files named file1.txt and file2.txt shown to the right. The contents of the file guess.c are shown on the left below:

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

int main(int argc, char **argv) {

    if (argc != 3) {
        printf("Usage\n");
        return 1;
    }

    int low = strtol(argv[1], NULL, 10);
    int high = strtol(argv[2], NULL, 10);
    int guess;
    scanf("%d", &guess);

    if (guess >= low && guess <= high) {
        printf("Correct\n");
    } else {
        printf("Incorrect\n");
    }
    return 0;
}
```

file1.txt:

5

file2.txt

2
4
6

Part (a) [3 MARKS] Assume the program guess.c has been compiled to produce an exectuable named guess. Write the output of the program (what is printed) for each of the following invocations:

`./guess 1 5 < file1.txt`

`./guess 2`

`./guess 5 10 < file2.txt`

Part (b) [1 MARK] The command sort has a flag -r that can be used to sort a file in reverse order. Use a combination of sort and guess to write a single unix command that invokes guess with command line arguments 5 and 10 and the last line of file2.txt as input.

Question 2. [3 MARKS]

Consider the following code fragments. Fill in the tables below with the values of the array elements at the point in the execution where the table appears. The first table is done for you.

```
int a[5] = {1, 2, 3, 4, 5};
```

a[0]	a[1]	a[2]	a[3]	a[4]
1	2	3	4	5

```
int *p = &a[1];
*p = a[0] + *p;
```

a[0]	a[1]	a[2]	a[3]	a[4]

```
p = a;
*(p + 2) += 1;
```

a[0]	a[1]	a[2]	a[3]	a[4]

```
p = a + 3;
*p = a[4];
```

a[0]	a[1]	a[2]	a[3]	a[4]

Question 3. [12 MARKS]

This question is based on the following course definition:

```
struct course {
    char *code;  // Points to a dynamically allocated string.
    int capacity;
    int num_enrolled;
};
```

Part (a) [4 MARKS] Complete the function `create_course` to create a new `struct course` with an initial enrolment of 0, the given capacity, and a dynamically allocated copy of the given code. The function must return a pointer to the new `struct course`.
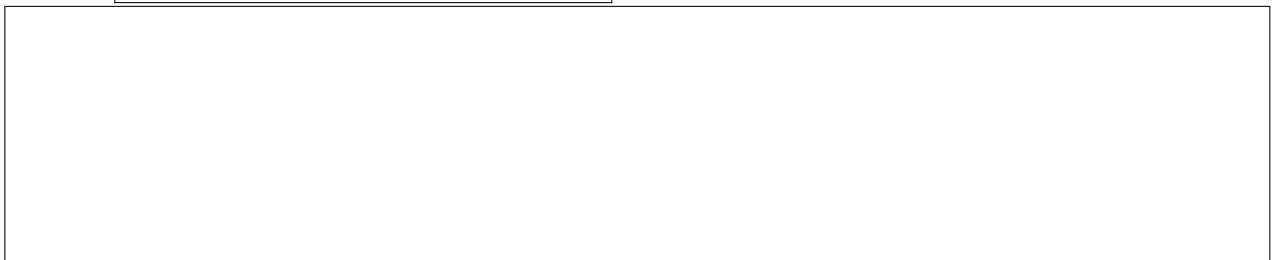
```
struct course *create_course(char *code, int capacity) {
```

```
}
```

Part (b) [2 MARKS] Complete the function `enrol` below. If the number of students enrolled in a given course is below the capacity, the function must increase the number of students enrolled in that course by one.

```
void enrol(                              ){
```
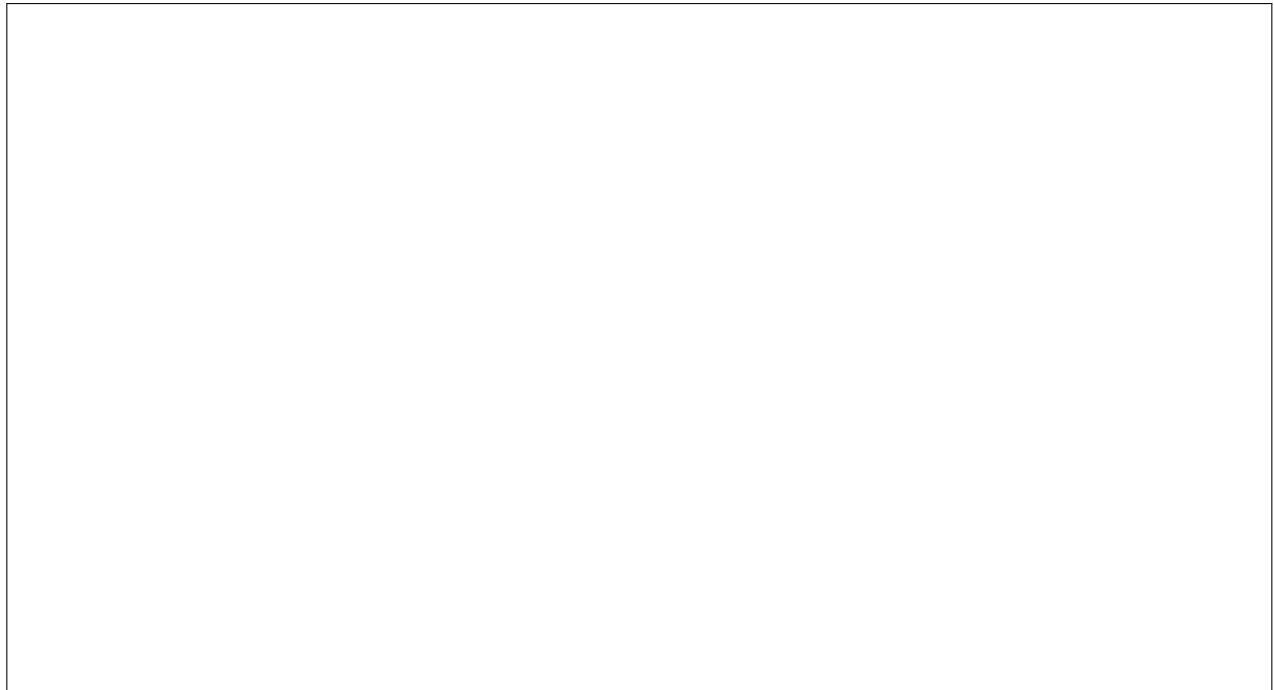
```
}
```

Part (c) [4 MARKS] A course code has the form "subject-number" (e.g., "CS-101", "MATH-9999", "BIO-50"). Complete the following function to return the subject (e.g., "CS", "MATH", "BIO") for the given course. The course (and its code) should not be modified. Allocate only as much memory as necessary.

```
char *get_subject(struct course c) {
```

```
}
```

Part (d) [2 MARKS] Consider the `main` function below. Add the code needed to free all dynamically-allocated memory for the program.

```
int main() {

    struct course *new_course = create_course("CSC-209", 500);

    char *subject = get_subject(*new_course);
    printf("The subject is: %s\n", subject);

    // Free memory



    return 0;
}
```

Question 4. [9 marks]

Part (a) [7 marks] Consider the code and memory diagram below. Fill in the memory diagram to show the current state of the program exactly before the return statement on **line 15** is executed. If there are uninitialized blocks of memory at that point in the program, write their values as ???. Label the stack frames with the corresponding function name.

```c
1   // Precondition: strlen(s) % n == 0 and n > 0
2   char *every_nth(char *s, int n) {
3
4     int size = strlen(s) / n;
5
6     char *result = malloc(sizeof(char) * size + 1);
7
8     int i = 0;
9     for(i = 0; i < size; i++) {
10      result[i] = s[i * n];
11    }
12
13    result[size] = '\0';
14
15    return result;
16
17  }
18
19
20  int main() {
21
22    char *input = "abcdefghi";
23
24    char *str = every_nth(input, 3);
25
26    printf("%s\n", str);
27
28    free(str);
29
30    return 0;
31  }
```

Section	Address	Value	Label
Read-only	0x100		
	0x104		
	0x108		
	0x10c		
	0x110		
	0x114		
	⋮	⋮	
Heap	0x23c		
	0x240		
	0x244		
	⋮	⋮	
Stack	0x454		
	0x458		
	0x45c		
	0x460		
	0x464		
	0x468		
	0x46c		
	0x460		
	0x474		
	0x478		
	0x47c		
	0x480		
	0x484		
	0x488		
	0x49c		

Part (b) [2 marks] If line 13 were omitted, the behaviour of the program would be undefined. Briefly explain what error could occur and why.

Error:

Why would that error occur?

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Total Marks = 28