Midterm Test — Sample Solutions

CSC209H1S / LEC0101/0201/L0301 — Campbell

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

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:

```
#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
```

Correct

```
./guess 2
```

Usage

```
./guess 5 10 < file2.txt
```

Incorrect

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.

```
sort -r file2.txt | ./guess 5 10
```

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

box 1 (given)

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

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

box 2

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

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

box 3

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

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

box 4

Midterm Test — Sample Solutions

CSC209H1S / LEC0101/0201/L0301 — Campbell

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

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) {

    struct course *new_course = malloc(sizeof(struct course));

    new_course->code = malloc(sizeof(char) * (strlen(code) + 1));
    strcpy(new_course->code, code);
    new_course->capacity = capacity;
    new_course->num_enrolled = 0;

    return new_course;
}
```

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(struct course *c) {
    if (c->num_enrolled < c->capacity) {
        c->num_enrolled++;
    }
}
```

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) {

    char *hyphen = strchr(c.code, '-');
    int size = strlen(c.code) - strlen(hyphen) + 1;

    char *subject = malloc(sizeof(char) * size);

    strncpy(subject, c.code, size-1);
    subject[size-1] = '\0';

    return subject;
}
```

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
    free(new_course->code);
    free(new_course);
    free(subject);

    return 0;
```

Midterm Test — Sample Solutions

CSC209H1S / LEC0101/0201/L0301 — Campbell

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

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.

```
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	abcd	
	0x104	efgh	
	0x108	i\0	
	0x10c		
	0x110		
	0x114		
	⋮	⋮	
Heap	0x23c	adg\0	
	0x240		
	0x244		
	⋮	⋮	
Stack	0x454		
	0x458		
	0x45c		
	0x460		
every_nth	0x464	0x100	`s`
	0x468		
	0x46c	3	`n`
	0x460	3	`size`
	0x474	0x23c	`result`
	0x478		
	0x47c	3	`i`
main	0x480	0x100	`input`
	0x484		
	0x488	???	`str`
	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: extra output or segmentation fault

Why would that error occur?

The string is not explicitly null-terminated.

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.