CMSC 15200 Midterm B

Lucius Gao

TOTAL POINTS

86 / 100

QUESTION 1

Short Answer 30 pts

1.1 a 7 / 8

- **0 pts** Correct

✓ **- 1 pts** **1 Wrong**

- **1 pts** Value and type flipped
- **2 pts** 2 wrong
- **4 pts** 4 wrong

1.2 b 12 / 12

✓ **- 0 pts** **Correct**

- **2 pts** Incorrect Answer
- **1 pts** Variable "da" is not drawn
- **0.5 pts** Minor wording mistake
- **1.5 pts** Not Entirely accurate

1.3 c 0 / 2

- **0 pts** Correct

✓ **- 2 pts** **not limit to main.c file**

- **2 pts** Wrong

1.4 d 2 / 2

✓ **- 0 pts** **Correct**

- **1 pts** should be "blah.h"
- **1 pts** no need for ;
- **0 pts** Click here to replace this description.

1.5 e 5 / 6

- **0 pts** Correct

✓ **- 1 pts** **Click here to replace this description.**

- **2 pts** Click here to replace this description.
- **3 pts** Click here to replace this description.
- **4 pts** Click here to replace this description.

QUESTION 2

Tracing Code 20 pts

2.1 a 9 / 10

- **0 pts** Correct

✓ **- 1 pts** **'\n' is not visible in output**

- **1 pts** There should be only one output
- **1 pts** Minor mistake
- **1 pts** should be left aligned
- **3 pts** Wrong # of lines
- **6 pts** Wong # of lines and # of * per line

2.2 b 7 / 10

- **0 pts** Correct
- **1 pts** 1 wrong output

✓ **- 2 pts** **2 Wrong output**

- **3 pts** 3 Wrong output
- **4 pts** 4 Wrong output
- **6 pts** 6 wrong output
- **1 pts** Should be integer type without floating point

✓ **- 1 pts** **\n is not visible**

- **1 pts** "" is not visible
- **1 pts** No line break

QUESTION 3

3 Iterative Coding 22 / 25

- **0 pts** Correct
- **1 pts** Critical Syntax Error
- **3 pts** Check for the NULL string
- **2 pts** Return after NULL string
- **3 pts** Incorrect use of pointers/dereferencing
- **1 pts** Use of two loops instead of one
- **2 pts** Update Counters Correctly
- **1 pts** Use of wrong type

✓ **- 2 pts** **First Initialization**

✓ **- 1 pts** **Second Initialization**

- **2 pts** Successful Return

- **5 pts** Loop Parts
- **3 pts** Algorithm Correctness
- **0 pts** Click here to replace this description.

QUESTION 4

4 Recursive Coding **22 / 25**

- **0 pts** Correct
- **2 pts** Checks arr[length] instead of arr[length-1]
- **7 pts** Does not return recursive case

✓ **- 2 pts** **Checks pointer instead of element at pointer**

✓ **- 1 pts** **Extra base case**

- **10 pts** No recursive case
- **2 pts** Recursive call with single element rather than entire array
- **1 pts** Syntax error
- **1 pts** Continues to perform recursive calls even if value is not positive
- **5 pts** No base case
- **2 pts** Returns 0 instead of 1 in base case
- **7 pts** Recursive statement never reached or used
- **3 pts** Incorrect base case check
- **5 pts** Incorrect base case
- **5 pts** Doesn't check current value (in case length = 1)
- **5 pts** Doesn't check current value
- **2 pts** Incorrect function call, no function name

Name: Lucios Gao

Lab Section time: 5:00

Exam 1
CMSC 152 11:30am
Winter 2020

Short Answer:	____/30
Tracing Code:	____/20
Iterative coding:	____/25
Recursive coding:	____/25
Total:	____/100

You are allowed one hand-written double-sided page of notes, and one reference sheet.
No electronic devices may be used at any time.
No one may leave the exam room and return to continue working on the exam.
No hats may be worn that obscure students' eyes

1. Short Answer (30 pts)

a. What is the result and type of the following expressions in C (2 pt each):

Expression	Value	Type
(char)('a'+4)	'e'	char
34/20	1	int
14 % 4	2	int
strlen("I can do this!!");	16	int

b. Answer the following questions about this line of code:

```
float da[] = {4.5, 7.3, -1.0, 10.3, 6.4, 7.9};
```

(3 pts) This allocation creates an array **consisting of** 6 **elements, each of type** float.

(2 pts) What is the value of da[4]? 6.4

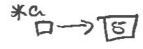
(2 pts) What is the value of *(da + 3)? 10.3

(2 pts) Draw a picture of what the memory looks like after the declaration & initialization line.

(3 pts) Explain the difference in how the asterisk is used for: `int *a;` **and** `*a = 5;`

int *a initializes a pointer of type int

*a=5 assigns whatever is at the address of what a points to to 5



c. (2 pts) How do you decide which functions to add to the .h file?

You add the prototypes of the functions written in the .c file that you want to be able to use in the main.c file

d. (2 pts) Write the line of code in blah_main.c that makes it read in the contents of blah.h

#include "blah.h"

e. (6 pts) Consider the functions foo and bar. bar calls foo.

```
void bar()
{
        int error, va = 3, wi = 1;
        int arr[] = {4, 7, 89, 5, -1, 0};
        int length = 6;
        error = foo(arr, length, &val, &wi);
}
```

```
void foo(int *ar, int *len, int v, int *w)
{
        *w = ar[(*len) - 2];
        v = 2;
        ar[v]++;
        *len = 5;
}
```

Identify the parameters as in parameter, out parameter, in/out both, or not a pointer. I give you v as an example.

ar	len	w	v
in	in/out both	out	*not a pointer*

2. Tracing Code (20 pts)

a. (10) Draw, in the right column, the output for the call print_shape(7)

```
void print_row(unsigned int n)
{     int i;
      for(i=0;i<n;i++)
            printf("*");
      printf("\n");
}

void print_shape(int n)
{
      if (n <= 0)
            return;
      print_row(n);
      print_shape(n-2);

}
```

Output

*******\n
*****\n
***\n
*\n

b. (10) Draw, in the right column, the output of this program.

```
void print(unsigned int n)
{     printf("%u\n",n);    }

int main()
{
      unsigned int m = 20, n = 140;
      unsigned int *pB = &n, *pA = &m;
      if (n > 100)
            print(n+38);
      else if (m < 50)
            print(n+10);
      else
            print(n+5);
      if (m < 40)
            print(n+100);
      if (n < 170)
            print(n);
      else
            print(n+17);
      print(*pA);
      print(*pB);
      *pB = 20;
      if (pA == pB)
            printf("==\n");
      if (*pA == *pB)
            printf("*==*\n");
      pA = pB;
      *pA = 15;
      print(m);
      print(n);
      print(*pA);
      print(*pB);
}
```

Output

178\n
20\n
140\n
==\n
20\n
15\n
15\n
15\n

3. Iterative Coding (25 pts)

```
/* min_max – a function that calculates the number of lowercase and uppercase letters in a string.
 * the results are placed in lower and upper. You may not call any functions within this code.
* If the string has length 0, then return 0. Otherwise, return 1.
* in parameters:
*   char *str – a pointer to the first character of the string
* out parameters:
*   int *lower – used to store the number of lowercase letters
*   int *upper – used to store the number of uppercase letters
* return value:
*   uint – 1 if success, 0 if the string length is 0
*/
typedef unsigned int uint;
uint lower_upper(char *str, uint *lower, uint *upper)
{  uint i;
    if (*str == '\0')
       return 0;
   for(i=0 ; *(str+i) != '\0' ; i++)
   {     //lower case case
         if ( *(str+i) >= 'a' && *(str+i) <= 'z')

            *lower ++;
         //upper case case
         else if ( *(str+i >= 'A' && *(str+i) <= 'Z')

            * upper ++;

    }

    return 1;
}
```

uint min_max??

Assuming the address these two pointers point to hold values of 0.

4. Recursive Coding (25 pts)

Write the function all_positive using recursion. This returns 1 if the array contains numbers that are all positive and 0 if it does not.

Input:	Output:
[]	1
[-4]	0
[3, 11, 9]	1
[3, -6, 9, 7]	0

arr[0] *(arr+0)

ptr[0] *(ptr+0)

```
typedef unsigned int uint;
uint all_positive(int arr[], uint length)
{
    // base case
    if (length == 0)
       return 1;
    if(length == 1)
    {
        if( arr[0] >0)
           return 1;
        else
           return 0;
     }
    //smaller case
    int s_case = all_positive( arr+1, length -1);
    //general case
    if( s_case ==1 && arr >0)
       return 1;
     else
       return 0;
}
```

3,11,9

cp(3,3)
↳ s_case = cp(11,2)
↳ s_case = cp(9,1) = 1

cp(9,1)
returns 1

cp(11,2)
s_case = 1
11 > 0
return 1

cp(3,3)
↳ s_case = 1
3 > 0
return 1