ENEE 140, Fall 2022
Final Exam — Answer Key
Do Not Make a Copy!!

Date:

University of Maryland Honor Pledge: The University is committed to Academic Integrity, and has a nationally recognized Honor Code, administered by the Student Honor Council. In an effort to affirm a community of trust, the Student Honor Council proposed and the University Senate approved an Honor Pledge. The University of Maryland Honor Pledge Reads:

"*I pledge on my honor that I have not given or received any unauthorized assistance on this examination (or assignment)*"

Please write the exact wording of the Pledge, followed by your signature, in the space below:

Pledge: _____
Pledge: _____
Pledge: _____
Pledge: _____

Your signature: _____
Full name: _____ Course: _____ Directory ID: _____

List of Exam Questions:

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	8	14	10	10	16	10	10	10	12	100
Score:										

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name, your section and your Directory ID on the front.
- Write your name and section at the bottom of each page as well.

- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 100 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. No electronic devices (e.g. laptops, tablets, smartphones, calculators) are allowed. Good luck!

1. (8 points) This question will test your knowledge structs and typedefs.

   Which of the following correctly creates a new data type student which contains a float field grade, string field name, which can be up to 80 characters long, and int field year.

   Note: You must be able to use this data type as follows

   ```
   student newStudent;
   ```

   A.
   ```
   struct student{
   int year;
   float grade;
   char name[80];
   };
   ```

   B.
   ```
   struct student{
   int year;
   float grade;
   char name;
   };
   ```

   C.
   ```
   typedef struct student{
   int year;
   float grade;
   char name[80];
   } student;
   ```

   D.
   ```
   typedef struct student{
   int year;
   float grade;
   char name[80];
   };
   ```

   Your answer:

   1. **C**

2. (14 points) This question will test your knowledge of 2-dimensional arrays.

   The following program is supposed to return the determinant of an array. You can assume the array passed in is a 2x2 array. Fill in the blanks.

   Note: The determinant of an array is calculated as $ad - bc$ where the array is $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

   ```
   int determinant(array[][]) {
       int det = array[ 0 ][ 0 ]*array[1][1] - array[0][ 1 ]*array[1][ 0 ];

       return det;
   }
   ```

Solution:

```
int determinant(int array[][]){

    int det = array[0][0]*array[1][1] - array[0][1]*array[1][0];
    return det;
}
```

3. (10 points) This problem tests your understanding of command line arguments and string library functions. Given the program below is compiled to the executable minmax, what is the output for commands listed below? **Note**: The program uses the `atoi()` function:

```
int atoi(char str[]);
//This function converts a string to an integer.
//Ex: atoi("123") returns 123.
```

The program:

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

int main(int argc, char* argv[])
{
    int result;
    if(argc < 3){
        printf("Usage: a.out <max, min> int (int) ...");
        return 0;
    }

    if(strcmp(argv[1], "max") == 0){
        result = INT_MIN;
        for(int i = 2; i < argc; i++){
            int tmp = atoi(argv[i]);
            if(tmp > result)
                result = tmp;
        }
        printf("The max value is: %d", result);
    } else if(strcmp(argv[1], "min") == 0) {
        result = INT_MAX;
        for(int i = 2; i < argc; i++){
            int tmp = atoi(argv[i]);
            if(tmp < result)
                result = tmp;
        }
        printf("The minimum value is: %d", result);
```

```
    } else
    {
        printf("Usage: a.out <max, min> int (int) ...");
    }

    return 0;
}
```

1. minmax max -3 4 5 8 3
2. minmax max
3. minmax min 1 1 1 1 1
4. minmax Max 4 5 6 -1 122
5. minmax min 5

Solution:

1. 8
2. Usage: a.out <max, min> int (int) ...
3. 1
4. Usage: a.out <max, min> int (int) ...
5. 5

4. (10 points) This question will test your understanding of functions and their syntax. The purpose of the **find_largest** function bellow is to loop through an array of 10 unsigned integers and return the largest value found. Fill in the blanks in the program snippet below such that the function compiles and works as intended.

```
_____ find_largest(_____ array[10]) {

    unsigned largest = 0;
    int i = 0;

    for(i = 0; i < 10; _____) {
        if(_____ > _____)
            largest = array[i];
    }

    return largest;
}
```

Solution:

```
unsigned find_largest(unsigned array[10]) {

    unsigned largest = 0;
    int i = 0;

    for(i = 0; i < 10; i++) {
        if(array[i] > largest)
            largest = array[i];
    }

    return largest;
}
```

5. (16 points) This question will test your understanding of arrays, loops, and functions. Determine the output to the following program:

```
#include <stdio.h>

int func(int x, int size, int a[]) {

    int sum = 0;

    for(int i = 0; i < size; i++) {
        a[i] += x++;
        sum += x;
    }

    return sum;
}


int main() {

    int sum, x = 3;
    int a[5] = {2, 4, 6, 3, 1};

    sum = func(x, 5, a);

    printf("sum is %d, x is %d\n", sum, x);
    printf("array a is %d %d %d %d %d\n", a[0], a[1], a[2], a[3], a[4]);

    return 0;
}
```

Solution:

```
sum is 30, x is 3
array a is 5 8 11 9 8
```

6. (10 points) This question will test your ability to read and understand code as well as debug it. The function cos(x) can be approximated by the following n degree Taylor series:

$$\sum_{k=0}^{n} \frac{(-1)^k x^{2k}}{(2k)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ...$$

Your friend wrote the following function to implement this n degree Taylor series for an input x, however they have not taken ENEE140 and so their code is riddled with errors. There are five errors on five lines. Find all five errors, and in the space bellow the function write the line number of each error along with the correct line.

Note: The inputs and outputs of cosine are both real-valued, and pow(a, b) returns a^b as a double.

```
1
2  #include <math.h>
3
4  float cos_taylor(int n, float x) {
5
6      int y = 0;
7      int fact;
8
9      for(int k = 0; k <= n; k--) {
10
11         fact = 0; //This loop determines (2k)!
12         for(int i = 1; i <= 2*k; i++) {
13             fact *= i;
14         }
15
16         //This if-else determines the alternating + and - from (-1)^k
17         if((k % 2) == 0)
18             y -= pow(x, 2*k)/(float)fact;
19
20         else if
21             y += pow(x, 2*k)/(float)fact;
22     }
23
24     return y;
25 }
```

Solution:

```
Line 6: float y = 0;
Line 9: for(int k = 0; k <= n; k++) {
Line 11: fact = 1;
```

```
Line 17: if(k % 2)
Line 20: else
```

7. (10 points) This question tests your ability to debug code.

   Dr. Dumitras wants to sort an array of 10 numbers using insertion sort and print out the original and sorted array. Unfortunately, he wrote his program late at night, and in his dreamlike fugue, made several mistakes in his implementation. Now, he looks to you to help him debug his code - you won't be paid of course.

   Find all of the bugs in the code and explain what went wrong. Be careful about making incorrect claims as Dr. Dumitras does not want to waste time fixing his program. For each incorrect bug you find, one point will taken away.

   Please write your final answers in the provided space below the program.

   Hint: There are 5 bugs for you to find.

```
1  #include <stdio.h>
2
3  /*
4   * Function will print out a passed-in array
5   */
6  void print_array(int array) {
7      int i;
8
9      printf("{");
10     for (i = 0; i < 9; i++) {
11         printf("%d, ", array[i]);
12     }
13     printf("%d}\n", array[i]);
14 }
15
16 int main() {
17     int original_array[10] = {20, 5, 11, -3, 0, 34, -12, 9, 100, 4};
18     int sorted_array[10] = original_array;
19     int value_to_insert, hole_position;
20     int i;
21
22     // For loop iterates across all elements of array and sorts the elements
23     for (i = 1; i < 11; i++) {
24         value_to_insert = sorted_array[i];
25         hole_position = i;
26
27         while ( (hole_position > 0) &&
28                 (sorted_array[hole_position-1] > value_to_insert) ) {
29             sorted_array[hole_position] = sorted_array[hole_position-1];
30             hole_position--;
31         }
32         sorted_array[hole_position] = value_to_insert;
33     }
34
```

```
35      printf("Original Array: ");
36      print_array(original_array[10]);
37
38      printf("Sorted Array: ");
39      print_array(sorted_array[10]);
40
41      return 0;
42  }
```

Solution:

```
Line 6 (3pts): void print_array(int array) {
        Argument should be "int array[]"
Line 18 (3pts): int sorted_array[10] = original_array;
        Assigning an array to another array is incorrect
Line 23 (2pts): for (i = 1; i < 11; i++) {
        Loop end condition incorrect: should be "i < 10"
Line 36 (1pt): print_array(original_array[10]);
        Argument should be "original_array"
Line 39 (1pt): print_array(sorted_array[10]);
        Argument should be "sorted_array"
```

8. (10 points) This program tests your understanding of function arguments. Write the output of the following program.

   Hint: What does pass-by-value mean?

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int special(int);
5  void specialWord(char [], int);
6
7  int main() {
8      int use = 0x7;
9      char word[60] = "electrical engineering";
10     special(use);
11     specialWord(word, 70);
12     printf("hello\n");
13     printf("Special number: %d\nSpecial Word: %s", use, word);
14 }
15
16 int special(int num) {
17     int num2 = num;
18     num2 = num2 >> 3;
19     num2 = num2 & 0xff;
20     num2 = (int) pow(((num2 << 1) >> 5) + (25 % num), 3);
21     return num2 + 1;
22 }
23
24 void specialWord (char word[], int max) {
25     int j;
26     for(j = 0; word[j] != '\0'; j++) {
27         if(j%2 && word[j] != ' ') {
28                 word[j] = word[j] + ('A' - 'a');
29         } else {
30                 word[j] = word[j];
31         }
32     }
33 }
```

Solution:

```
hello
Special number: 7
Special Word: eLeCtRiCaL EnGiNeErInG
```

9. (12 points) This question tests your understanding of control and conditional logic. In the program below, the variable `FLUBBER` is back for round two and is eager for another ride. Write down the outputs of this program.

   To do this, you should track the progress of `FLUBBER` across the program, keep in mind all changes to its value, and write down what what each of the `printf` statements will print out.

   Please write your final answers in the designated section below the program.

```
1  #include <stdio.h>
2
3  int mystery_function() {
4      static int FLUBBER = 0;
5      FLUBBER = (FLUBBER > 5) ? (FLUBBER+1) : (FLUBBER+2);
6      return FLUBBER;
7  }
8
9  int main() {
10     int FLUBBER;
11     int i;
12
13     for (i = 0; i < 5; i++) {
14         FLUBBER = mystery_function();
15     }
16     printf("FLUBBER = %d\n", FLUBBER);
17
18     switch(FLUBBER % 2) {
19         case (0):
20             FLUBBER *= 2;
21         case (1):
22             FLUBBER *= 3;
23         default:
24             FLUBBER += 18;
25             break;
26     }
27     printf("FLUBBER = %d\n", FLUBBER);
28
29     if (FLUBBER % 2 == 0) {
30         FLUBBER += 1;
31     } if (FLUBBER % 2 == 1) {
32         FLUBBER += 1;
33     }
34     printf("FLUBBER = %d\n", FLUBBER);
35
36     // PROGRAM CONTINUES ON NEXT PAGE
37
```

```
38      for (i = 0; i < 3; i++) {
39          if (i == 2)
40              continue;
41          FLUBBER += i;
42      }
43      printf("FLUBBER = %d\n", FLUBBER);
44
45      return 0;
46  }
```

Solution:

LN16: FLUBBER = 8
LN27: FLUBBER = 66
LN34: FLUBBER = 68
LN43: FLUBBER = 69