ENEE 140, Fall 2023
Final Exam — Answer Key
Do Not Make a Copy!!

Date: Thursday, December 14, 2023, 8–10 am

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:	16	8	4	8	15	10	16	17	6	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. (16 points) This problem tests your understanding of C types and casts and of C operators. Assume that variables `a`, `b`, `c` and `d` are defined as follows:

```
float a = 3.0;
int b = 6;
char c = 'D';
unsigned d = 5;
```

Fill in all the empty cells in the table below. For each of the C assignment expressions in the left column, state the resulting value of the `r2`–`r9` variables. If an expression results in a compilation error, write ERROR.

Assignment		Value
float	r1 = b / d;	1.0
float	r2 = d / b * b;	**0.0**
unsigned	r3 = b % (b / a);	**ERROR**
int	r4 = (**int**)a << (b − d);	**6**
unsigned	r5 = 2 * UINT_MAX + 6;	**4**
char	r6 = c + b;	**'J'**
int	r7 = (d < b);	**1**
int	r8 = (b & 1) && 1;	**0**
unsigned	r9 = d % (−−b − d);	**ERROR**

2. (8 points) This question tests your ability to correctly identify string declarations in C. Identify for each declaration if it is True (correct) or False (incorrect) in the box to the right of the declaration

a. `char string1[] = "Hello, World!";`	
b. `char string2[];`	
c. `char string3[10] = {'H', 'e', 'l', 'l', 'o', '\0'};`	
d. `char string4[][] = "Hello, World!";`	

Solution:

a. True
b. False
c. True
d. False

3. (4 points) This problem test your understanding of bitwise operators.

In C, you are able to take a number and manipulate its bits utilizing bitwise operators including, but not limited to, `&`,`|`,`<<`. Which of the following bitwise operations will give you the value of b_7 of the variable `'a'`?

A. a & 128

B. (a >> 7) | 1

C. a | 7

D. (a >> 7) & 1

3. **D**

4. (8 points) This question will test your understanding of structs and arrays. The following code snippet describes two structs that can be used to store information about a class of students.

```
typedef struct _book {
    int pages;
    float price;
    char title[100];
} Book;

typedef struct _bookstore {
    Book catalog[20];
    char store_name[100]
} Bookstore;

int main(){
    Bookstore hornbake;

    ...

}
```

Note: In the code above, `'hornbake.catalog[4].pages'` holds the number of pages in the 5th book.

(a) Which variable corresponds to the store name of the bookstore?

A. hornbake.store_name

B. hornbake.store_name[0]

C. hornbake.catalog.store_name

D. bookstore.store_name

(a) A

(b) Which variable corresponds to the string that holds the title of the book in position n?

A. hornbake.catalog[n]. title [n]

B. hornbake.book[n]

C. hornbake.catalog[n]. title

D. hornbake.catalog[n]

(b) C

(c) Which variable name corresponds to the price of the nth book in the catalog?

A. hornbake.catalog.price[n]

B. hornbake[n].price

C. hornbake[n].catalog.price

D. hornbake.catalog[n].price

(c) D

(d) Which variable name corresponds to the first letter of the third book in the catalog?

A. hornbake.catalog[0].title[3]

B. hornbake.catalog[2].title[0]

C. hornbake.catalog[3]

D. hornbake.catalog[1].title[3]

(d) B

5. (15 points) For each of the following questions, only one answer is correct. Indicate which one.

(a) If integer `"a"` has initial value `10`, what is the new value of `"a"` after the following operation:

```
a = a & 1
```

A. 0

B. 1

C. 5

D. 10

(a) A

(b) If integer `"b"` has initial value `12`, what is the new value of `"b"` after the following operation:

```
b = b | 6
```

A. 0

B. 1

C. 4

D. 14

(b) D

(c) If integer "c" has value 8, what will be the value of integer "d" after the following operation:

```
d = c >> 2
```

A. 1

B. 2

C. 4

D. 32

(c) B

(d) Given the following code segment, which of the below options is the correct conditional expression?

```
if (x <= 10 ) {
    y = 1;
} else {
    y = 2;
}
```

A. y = (x <= 10) : 1 ? 2

B. y = (x <= 10) : 2 ? 1

C. y = (x <= 10) ? 1 : 2

D. y = (x <= 10) ? 2 : 1

(d) C

(e) Suppose you are given a compiled program called "my_program". You are told to run "my_program" using the following command line text:

```
./my_program 21
```

Based on what you know, which of the following options is correct?

A. argc == 1, argv[0] is "21"

B. argc == 1, argv[1] is "21"

C. argc == 2, argv[0] is "21"

D. argc == 2, argv[1] is "21"

(e) D

6. (10 points) This problem tests your understanding of the switch statement and cases. Convert the code segment below from using if/else to using switch and cases.

```
if (num == 0 || num == 1 || num == 2 || num == 3) {
    printf("Number is less than 4\n");
} else if (num == 4) {
    printf("Number is 4\n");
} else if (num == 5) {
    printf("Number is 5\n");
} else {
    printf("None\n");
}
```

Please continue the below code.

```
switch (num) {
```

Solution:

```
switch (num) {
    case 0:
    case 1:
    case 2:
    case 3:
        printf("Number is less than 4\n");
        break;
    case 4:
        printf("Number is 4\n");
        break;
    case 5:
        printf("Number is 5\n");
        break;
    default:
        printf("None\n");
        break;
}
```

7. (16 points) This problem tests your understanding of bit masking.

   You've been hired as a spacecraft avionics engineer at a contracting laboratory to help develop NASA's upcoming Dragonfly rotor-craft mission to Titan. Experienced with building resilient systems, you recognize the risk of cosmic ray "bit-flips" to critical onboard electronics during prolonged radiation exposure on interplanetary missions. If unprotected for, such an error could be disastrous for the $300-million program.

   A quick method to check for "bit-flips" is to implement a bit parity (even/odd) error detector. In short, **if a binary transmission sequence is expected to contain an even number of 1's** (`parity = 0`)**, a bit flip error would cause the number of 1's in the sequence to switch parity and become odd.**

   For example, if the transmitted data is $(16)_{10}$, your expected `parity` would be 1 since there are an odd number of 1's in the binary representation of `data`.

   Your task is to **complete the following bit flip detection function.** Your binary sequence will be passed in as an integer called `data`. You MUST use all predefined macros in your implementation; return PASS if the bit parity is correct, return BIT_ERROR if not.

```
#define PASS 0
#define BIT_ERROR 1

int check_parity_bit(int data, int data_size, int parity) {
    int i;
    int count = 0;

    // Count the number of set bits (1's) in the data
    for (i = 0; i < data_size; i++) {
        if ((  data  & (  1  << i )) != 0 ) {
            count ++;
        }
    }

    // Check if the bit parity is correct
    return (count % 2 == parity) ? PASS : BIT_ERROR ;
}
```

Solution:

```
#define PASS 0
#define BIT_ERROR 1

int check_parity_bit(int data, int data_size, int parity) {
    int i;
    int count = 0;
```

```
    // Count the number of set bits (1's) in the data
    for (i = 0; i < data_size; i++) {
        if ((data & (1 << i)) != 0) {
            count++;
        }
    }

    // Check if the bit parity is correct
    return (count % 2 == parity) ? PASS : BIT_ERROR;
}
```

8. (17 points) This question tests your ability to open a file, read the file line by line, then write to a separate file. Assume that you are given two files "input.txt" and "output.txt" through the command line. You may assume that argv[1] = "input.txt" and argv[2] = "output.txt" and that they will always be there. Your task is to read through each line of "input.txt" and put all of the letters (no numbers, whitespace or any other types of special characters) into "output.txt". Below is an example for the input and output text files.

"input.txt":	"output.txt":
Hello World 123!	HelloWorld
This is a sample text.	Thisisasampletext
Testing 1, 2, 3.	Testing

File in the blanks for the following code

```
#include <stdio.h>
#define MAX_STRING 1000

int main(int argc, char *argv[]) {
    FILE *input;
    FILE *output;
    char line[MAX_STRING];
    int i;

    // open files
    input = fopen(_____, _____);
    output = fopen(_____, _____);

    // reading each line
    while(fgets(_____, _____, _____)){
        i = _____;
        // scanning the line for each letter
        while(line[i]!=_____){
            if((_____ >= 'A' && _____) ||
               (_____ && _____)){

                fputc(_____, _____); // put character in output

            }
            i++;
        }

        // add newline to output
        fputc(_____, _____);
    }

    return 0;
}
```

Solution:

```
#include <stdio.h>
#define MAX_STRING 1000

int main(int argc, char *argv[]) {
    FILE *input;
    FILE *output;
    char line[MAX_STRING];
    int i;

    // open files
    input = fopen(argv[1], "r");
    output = fopen(argv[2], "w");

    // reading each line
    while(fgets(line, MAX_STRING, input) != NULL){
        i = 0;
        // scanning the line for each letter
        while(line[i]!='\0'){
            if((line[i] >= 'A' && line[i] <='Z') ||
               (line[i] >= 'a' && line[i] <= 'z')){

                fputc(line[i], output); // put character in output

            }
            i++;
        }

        // add newline to output
        fputc('\n', output);
    }

    return 0;
}
```

9. (6 points) This question has two parts. It tests your understanding of the break and continue keywords in C. Look at the following snippets if code and determine how many times "HELLO" is printed.

(a)

```
#include <stdio.h>

int main() {
```

```
    int i, j;

    for (i = 1; i <= 3; i++) {
        printf("HELLO\n");

        for (j = 1; j <= 5; j++) {
            if (j % 2 == 0) {
                printf("HELLO\n");
                continue;
            }
        }
    }

    return 0;
}
```

(a) 9

(This question continues on the next page)

(b)

```c
#include <stdio.h>

int main() {
    int i, j;

    for (i = 1; i <= 3; i++) {
        printf("HELLO\n");

        for (j = 1; j <= 5; j++) {
            if (j % 2 == 0) {
                printf("HELLO\n");
                continue;
            }

            if (i == 2 && j == 3) {
                break;
            }
        }
    }

    return 0;
}
```

(b) 8