# ENEE 140, Fall 2024
# Final Exam

**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 | 10 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Points: | 12 | 8 | 12 | 9 | 12 | 12 | 14 | 10 | 11 | 12 | 112 |
| 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 last question is for EXTRA CREDIT and is worth 12 points. You may receive full credit without answering it, assuming that you answered correctly all the other questions (the exam will be scored out 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. (12 points) This is a four-part, multiple-choice question.

   (a) What is the min value that can be stored by the int datatype in C, assuming w = 32?

   A. $-0$

   B. 255

   C. $-2^{31} - 1$

   D. $-2^{w-1}$

   (a) ______________

   (b) Which keyword is used to specify that a function does not return any value in C?

   A. **int**

   B. **float**

   C. **void**

   D. **char**

   (b) ______________

   (c) What is the output of the following C code?

```
#include <stdio.h>
int main() {
    int a = 10;
    printf("%d", a--);
    printf("%d", a);
    return 0;
}
```

   A. 1010

   B. 109

   C. 99

   D. 119

   (c) ______________

(d) How do you declare a const variable in C that holds a floating-point value?

A. All of the below

B. **float const** x = 3.14;

C. **#define** PI 3.14;

D. **const float** x = 3.14;

(d) ______________

2. (8 points) The following code tests your understanding of command line arguments. It first checks that enough arguments were entered, then prints the arguments entered, and finally checks if the first character of the last command line argument string is a number. If we were to compile this code with **a.out hello 4** it would output:

```
a.out hello 4
The first character of the third argument is a number.
```

Fill in the corresponding blanks to complete the program.

```
#include <stdio.h>

int main(int argc, char *argv[]){
    int x = 0;

    if (argc ____________ 3){
        fprintf(stderr, "%d is too few arguments.\n", argc);
        return 0;
    }

    while (x < ____________){
        printf("%s ", argv[x++]);
    }

    if (____________ >= '0' && ____________ <= '9'){
        printf("\nThe first character of the third argument is a number.");
    }

    return 0;
}
```

3. (12 points) This question will test your understanding of do-while loops, variables, the modulus operator. What is the output of the following code?

```
int x = 5;
int y = 2;

do {
     printf("x is: %d\n",x++);
     if ((x % 4) == 0){
          printf("x/y is: %.1f\n", (float)x/y);
     }
} while (x < 10);
```

4. (9 points) This problem will test your knowledge on loops. Given the below while loop:

```
int i = 0;
while(i < 5) {
     i++;
     printf("%d\n", i);
}
```

Fill in the blank spaces of the below for loop in order to get the same output as the above while loop.

```
for(int i = ________; ________; ________){
     printf("%d\n", i);
}
```

5. (12 points) This question will test your understanding of `switch` statements. Convert the code segment below from using `switch` statements to using `if/else if/else/` statements. Your answer **should only** include code **pertaining to the segment below** and nothing else: **don't include** statements for header files, `main`, variable declarations, etc.

```
 switch (number) {
    case 0:
    case 2:
    case 4:
    case 5:
        printf("The value of number is %d.\n", number);
    case 7:
    case 58:
        printf("The value of number is extremely random!\n");
        break;
    case 47:
        printf("There is no need for this specific number.\n");
        break;
    default:
        printf("The number is irrelevant.\n");
        break;
}
```

6. (12 points) This problem will test your knowledge on variable scope. What is the output of the following program?

```
#include <stdio.h>
int fn() {
    static int i = 0;
    i += 2;
    return i;
}
int main(){
    for(int i = 0; i < 4; i++){
        printf("%d\n", fn());
    }
    return 0;
}
```

______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

7. (14 points) This problem will test your understanding of functions, loops, and number manipulation. This program intends to reverse a given integer. For example, an entry of 123 should print out 321. Fill in the blanks within `reverseNumber()` function to make this occur.

The purpose of the `reverseNumber()` function is to reverse the digits of the given number.

```
int reverseNumber(int num) {

    int reversed = ____________;

    while (____________) {

        int remainder = ____________;

        reversed = (____________ * 10 ) + ____________;

        num = ____________;

    }

    return ____________;
}
```

8. (10 points) This question will test your ability to understand the `struct` data type and how to implement and use it. The program below uses a `struct` data type to record the full name, tickets purchased, and ticket cost of a certain number of students at UMD. Fill in the blanks to make the program work properly. Assume that Student #0 is the **beginning** "student" (**element**) of the struct array, Student #1 is the next one, and so on.

```
#include <stdio.h>
#define STUDENTSIZE 30
#define MAXNAME 80

struct student {
    char firstname[MAXNAME];
    char lastname[MAXNAME];
    int ticketspurchased; /* amount of tickets purchased per EACH student */
    float ticketcost; /* order (price) total per EACH student profile */
};

int main() {
    ____________ student UMD[STUDENTSIZE];
    int total_tickets = 0; /* amount of tickets purchased for ALL students */
    float total_cost = 0.0; /* order (price) total of ALL students */

    /* Enter Information into Data Struct */
    for (int i = ____________; i < ____________; ____________) {
        printf("Enter First and Last Name of Student #%d:\n", ____________);

        scanf("%____________", ____________);
        scanf("%____________", ____________);

        printf("Enter Student #%d's Tickets Purchased/Cost:\n", ____________);

        scanf("%____________", ____________);
        scanf("%____________", ____________);
    }

    /* Calculate Total Amount of ALL Tickets Purchased and Cost */
   for (int i = ____________; i < ____________; ____________) {
        total_tickets ____________;
        total_cost ____________;
    }

    /* Print Beginning Letter of Student #4's Last Name */
    printf("%____________", ____________);

   return 0; }
```

9. (11 points) This two-part question tests your understanding of sorting. Examine the code below.

```
int tmp, i, j;
for (i=0; i<n; i++) {
    for (j=i+1; j<n; j++) {
        if (a[i] > a[j]) {
            tmp = a[i];
            a[i] = a[j];
            a[j] = tmp;
        }
    }
}
```

(a) Which sorting algorithm is this?

A. Bubble sort

B. Selection sort

C. Insertion sort

D. Quick sort

(a) ______________

(b) Write the resulting array after each iteration of the outer loop when this algorithm is performed on {20, -3, 99, 2, 31}.

10. (12 points) EXTRA CREDIT This question tests your understanding of C variables and operations. Your goal is track the variable known as POLANDIUM as it progresses through the code. Note the changes to its value, and write down what each of the printf statements will print out. Don't lose it, keep watch of it!

```c
#include <stdio.h>

int earth(int a);
int siberia(int b);
void climate(int c);

int main() {
    int POLANDIUM = 140;
    printf("POLANDIUM = %d\n", POLANDIUM);

    POLANDIUM = earth(POLANDIUM);
    printf("POLANDIUM = %d\n", POLANDIUM);

    POLANDIUM += 0x63;
    printf("POLANDIUM = %d\n", POLANDIUM);

    climate(POLANDIUM);
    printf("POLANDIUM = %d\n", POLANDIUM);

    POLANDIUM += ((10 & 7) / 2) + (1 & 1) - 1;
    printf("POLANDIUM = %d\n", POLANDIUM);

    POLANDIUM = siberia(POLANDIUM);
    printf("POLANDIUM = %d\n", POLANDIUM);

    return 0;
}

int earth(int a) {
    int continent;
    int ocean = 0;

    for (continent = 0; continent < 3; continent++) {
        ocean += continent * 3;
    }

    if (ocean > 3) {
        return a + 1;
    } else {
        return a + 2;
    }
}
```

```c
void climate(int c) {
    int weather = 0;
    int terrain = c + 1;

    do {
        weather += terrain + weather;
        terrain = weather % 30;
    } while (weather % terrain != 0 && terrain != 0);
}

int siberia(int b) {
    int snow;
    int ice = 0;

    for (snow = 2; snow > 0; snow--) {
        ice += (snow * 2);
    }

    if (ice >= 6) {
        b = 246;
        return b;
    } else {
        b = 461;
        return b;
    }
}
```