Name: JHEID: Score:

Homework 2

⚠ CAUTION

- You are expected to work individually.
- **Due: Friday September 25ᵗʰ at 11pm EDT (Baltimore time).**
- *This assignment is worth 20 points.*

💣 SUBMISSION REQUIREMENT

Answer each problem in this **pdf**, in the area to the side of the problem, or immediately after it. You may either type your solutions, or hand-write and scan them in, but they need to be legible, and part of this document. If you need to add additional sheets, please make a note near the problem itself that the grader should "see attached". Submit the **pdf** document via GradeScope once you have added your answers.

Learning Objectives

💡 OBJECTIVES

- control flow
- c-style strings
- data types
- arrays

ℹ INFO

Many problems make use of "code fragments", which can be thought of as pieces of code extracted from complete programs. While a code fragment will not generally compile by itself, we will assume that it exists in a sensible framework (i.e. is inside a properly formed `main()`, all appropriate headers and libraries have been included, etc.). We will also assume that there is no other code in the program that would impact the behavior of the fragment; each fragment is designed to be understood in isolation.

Part I: Code Puzzles. [1 point each problem]

Trace through each code fragment and write down the exact output that will be printed if the fragment is run, assuming it is embedded in a proper program with the necessary **#include** statements. If there is no output generated, write "**no output**", and give one sentence explaining why.

> 💡 **TIP**
>
> Note that these are called "puzzles" because their behavior may not be intuitive or correct (though the code itself is valid and will compile, albeit with warnings in some cases). *If you think you have spotted a typo in Part I, it is intentional!*

1. ```c
   int i = 1;
   while (i < 10); {
       if (i % 2 >= 0)
       printf("%d ", i++);
   }
   ```

> 🖋 **ANSWER:**

2. ```c
   for (int i = 0 ; i < 5 ; i++) {
       for (int j = 0 ; j < 2 ; j++) {
           if( i == j + 1 ) { break; }
           else { printf( "%d %d\n" , i , j ); }
       }
   }
   ```

> 🖋 **ANSWER:**

3.
```c
int x = 0;
while (true) {
    printf("x = %d\n", ++x);
    if (x = 4) { break; }
}
```
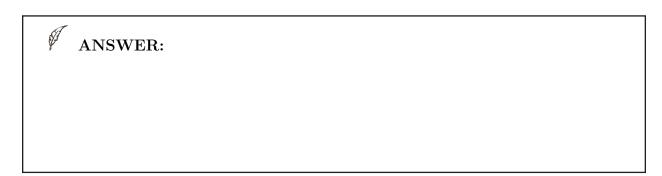
ANSWER:

4.
```c
int i = 15;
while (i > 10) {
    int sum = i;
    sum = sum + i;
    printf ("%d\n", sum);
    i--;
}
```

ANSWER:

5.
```c
char str[] = "this is a test only!";
for(int i = 0; i < (int)strlen(str); i++) {
    printf("str[%d] = %c\n", i, str[i]);
    if(str[i] == ' ') { str[i] = 0; }
}
printf("strlen( %s ) = %d\n", str, (int)strlen(str));
```

ANSWER:

6.
```c
char c = 'A';
while (c >= 'a')
    printf("%c ", c--);
```

ANSWER:

7.
```c
int a = 1;
switch(a) {
    case '1':
        printf("ONE\n");
    break;
    case '2':
        printf("TWO\n");
    break;
    defaultt:
        printf("THREE\n");
}
```

ANSWER:

8.
```c
float f = 0.0f;
int i;
for(i = 0; i < 20; i++)
    f = f + 0.1f;
if (f == 2.0f)
    printf("f is 2.0 \n");
else
    printf("f is NOT 2.0\n");
```

✍ **ANSWER:**

9.
```c
int i = 3;
if ((--i < 3) || (i--/4) || !(i-- > 2))
    printf("Hello\n");
if (i--)
    printf("Goodbye\n");
printf("%d\n", i);
```
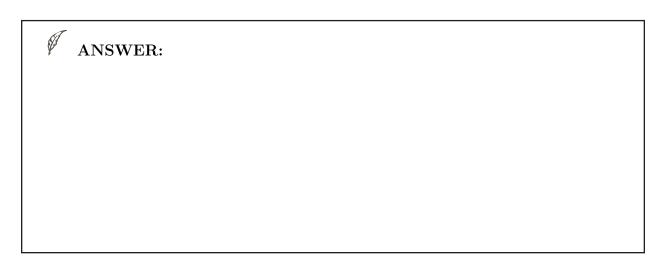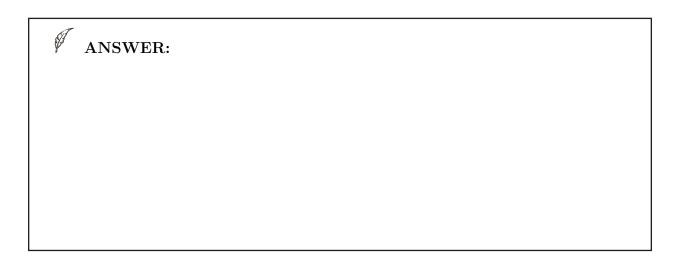
✍ **ANSWER:**

Part II: Code Correctness. [1 point each problem]

Trace through the code fragments and explain what is wrong with them. You are not expected to show the output.
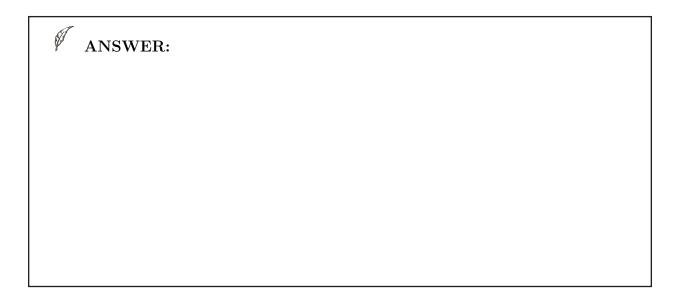
10.
```c
// collect a valid face number of a deck
int face = 0;
do {
    char msg[] = "Please enter the face number [1-13]: ";
    printf( "%s", msg);
    scanf("%d", &face);
} while (face >= 1 && face <= 13);
```

> ANSWER:

11.
```c
float n;
printf("Enter a number: ");
scanf("%f", n);
printf("You entered %f \n", n);
```

> ANSWER:

12.
```c
char source[] = "hello folks";
char destination[11];
strcpy(destination, source);
for(int i = 0; source[i]; i++) {
    printf("%c" , source[i]);
}
printf("\n");
for(int i = 0; destination[i]; i++) {
    printf("%c" , destination[i]);
}
printf("\n");
```

✎ **ANSWER:**

Part III: Code Reading. [2 points each problem]

At a high level, explain what the following functions or code fragments do. The explanation should **not** be a direct translation of the code statements.
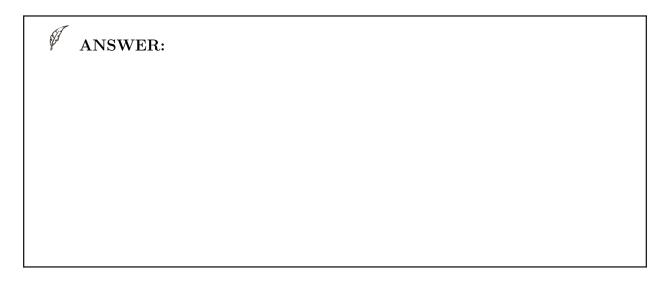
13.
```c
unsigned fun1(unsigned a, unsigned b) {
    int count = 0;
    int sum = b;
    while (sum <= a) {
        sum += b;
        ++count;
    }
    return count;
}
```

> ✒ **ANSWER:**

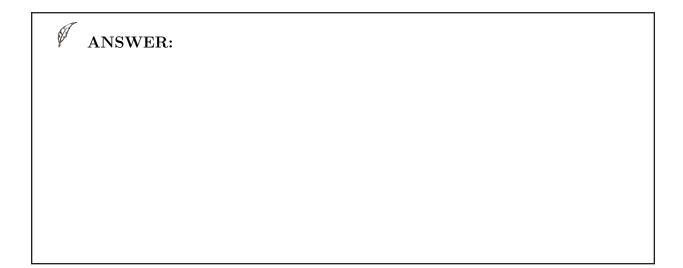14.
```c
// Assume that "str" is assigned some string value
void fun2(char str[]) {
    for(unsigned int i = 0; i < strlen(str); i++) {
        if( str[i] >= 'A' && str[i] <= 'Z' ) {
            str[i] -= 'A' - 'a';
        }
    }
    printf("%s\n", str);
}
```

> ✒ **ANSWER:**

15.
```c
int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int p = sizeof(arr) / sizeof(int) - 1;
int k = arr[p];
for (int j = p; j >= 1; j--)
    arr[j] = arr[j - 1];
arr[0] = k;
```

✏ **ANSWER:**

16.
```
// Assume "abs" is a defined/accessible function and
// returns the absolute value of what is passed into it
int fun3(int a, int b) {
    return ((a + b) + abs(b - a)) / 2;
}
```

✏ **ANSWER:**

<The END of Homework 2>