ENEE 140, Spring 2020
Midterm Exam — Answer Key
Do Not Make a Copy!!

Q2

a)

```
i = 0;
while(num > 0){
        a[i] = num%4;
        num = num/4;
        i++;
}
```

b) This program prints out the base 4 equivalent of a user inputted number (in base 10). It is displayed in the “correct” order, where the Most Significant Bit is the leftmost digit and the Least Significant Bit is the rightmost digit.

Q3

1) no typecasting
2) d is cast implicitly to unsigned, then expression is cast explicitly to int when assigned to num2
3) the expression is cast implicitly to float when assigned to num3
4) d is cast explicitly to float. expression is cast implicitly to unsigned when assigned to num4
5) b is cast implicitly to double, then expression is cast implicitly to int when assigned to num5
6) expression is cast implicitly to double when assigned to num6

Q4

Based on the snippet of code, there are 3 equations that are presented (and 3 unknown variables). These equations are:

```
6 = x + y;
7.5 = (z+x+y)/3;
3 = 7.5/x;
```

The first step is to solve for x. Since 7.5/x = 3, and r3 is an unsigned integer, the value of x must be 2 (if x were 3, r3 would equal 2. No other values for x are plausible). Plugging in the value of

x to the first equation yields 6 = 2 + y, or y = 4. Plugging in the values of x and y to the second equation yields 7.5 = (6+z)/3, or z = 16.50. Z is a float, so there is no casting involved for the result.

x = [2]
y = [4]
z = [16.5 (or equivalent)]

Q5

Line 9: int arraySize, limit, count;
Line 13: printf("Enter the size of array\n");
Line 14: scanf("%d", &arraySize);
Line 22: while(count < arraySize)
Line 27: printArray(array, arraySize);
Line 33: return 0;
Line 45: printf statement not needed, can be removed.
Line 45: i++;

Q6

```
char get_test_result(unsigned UID, char class_code[], int class_number,
                     unsigned test_number);
```

Q7

```
printf("UID: %u\n%s%d exam %u score: %c", UID, class_code, class_number,
        test_number, grade);
```

Q8

There are obviously a variety of correct approaches to the problem but some things to check when grading to ensure the user's solution is correct -

1. See if the program counts 0 as a number that is evenly divisible by the user's input (it should NOT)
2. Check the input code to make sure the student is using scanf for an int or using getchar and then converting to an int with atoi (or using scanf and then converting to an int, though that's the least efficient way)
3. The formatting of the user's output printf statements shouldn't be a cause for removing points unless it's so bad that it would obscure the meaning of the output (ie code that has no newlines and no labelling of the values so that they run together would warrant some points off but basically nothing shy of that)

Here's the simplest way to solve it:

```
#include <stdio.h>

int main() {
```

```
    printf("Please enter a positive integer:");
    int input = 0, total = 0, i = 0;
    scanf("%d", &input);

    for (i = 1; i <= input; ++i) {
        if (i % 5 == 0) {
            ++total;
        }
    }

    printf("\nThe total of the divisible numbers is %d\n", total);

    return 0;
}
```

Q9

The program needs to update the value of the user input with getchar() inside the while loop. Otherwise, nothing changes and the while loop will infinitely loop, printing the error message until the program crashes.

```
#include <stdio.h>

char get_input();

int main() {
    printf("The user's inputted letter is %c\n", get_input());
    return 0;
}

char get_input() {
    printf("Please enter a letter: \n");
    char input = getchar();

    while (!(input >= 'a' && input <= 'z') && !(input >= 'A' && input <= 'Z')) {
        printf("You did not enter a letter! Try again\n");
        input = getchar(); //this is the line that needs to be added
    }

    return input;
}
```