MTH 307: Programming and Data Structures

Practice Assignment I

1. What is the output produced by each of the following program fragments.

   (a)
   ```
   for(i = 5, j = i - 1; i > 0, j > 0; --i, j = i - 1)
   printf("%d",i);
   ```

   (b)
   ```
   for(i = 10; i >= 1; i /= 2);
   printf("%d", i++);
   ```

   (c)
   ```
   i = 9384;
   do
   {
   printf("%d",i);
   i /= 10;
   } while (i > 0);
   ```

   (d)
   ```
   i = 1;
   while (i <= 128)
   {
   printf("%d",i);
   i *= 2;
   }
   ```

   (e)
   ```
   sum = 0;
   for(i = 0; i < 10; i++)
   {
   if (i % 2) continue;
   sum += i;
   }
   ```

2. Write C programs to execute the following tasks.

   (a) Writing a function so that when an array `a` of length `n` is passed, the function will search for largest and smallest elements in `a` and store them in the variable pointed to be `largest` and `smallest`. For example,

   ```
   void find_largest_smallest(int a[], int n, int *largest, int *smallest)
   ```

   (b) Computing the sum of the elements of a two-dimensional array of numbers using pointer arithmetic. Please note that only one loop should be used.

   (c) Reversing a string and determining whether it is a palindrome by using a pointer to keep track of array subscripting.

   (d) Declaring a structure tag named `complex` with two arguments `real` and `imaginary` of type `double` for the following purposes.

      (i) Passing a variable of `complex` type into separate functions for calculating and returning the inverse, modulus, and conjugate of a complex number.

      (ii) Passing two variables of `complex` type into separate functions for adding, subtracting, multiplying, and dividing two complex numbers, and then returning resulting complex number of `complex` type.

(e) Declaring a structure tag named `fraction` with two arguments `numerator` and `denominator` of type `long` for the following purposes.

   (i) Passing a variable of `fraction` type into a function for calculating and returning the resulting fraction in reduced form.

   (ii) Passing two variables of `fraction` type into separate functions for adding, subtracting, multiplying, and dividing two fractions, and then return resulting fraction of `fraction` type in the reduced form.