using System;  
  
  
namespace Leavines\_IEEE  
{  
    class MainClass  
    {  
        public static void Main(string[] args)  
        {  
            Console.WriteLine("Please enter your Decimal Floating Point: "); // Prompts user to input  
            decimal x = decimal.Parse(Console.ReadLine()); // Gets user input  
            int bit; // Sign bit is assigned   
            int mult = 0; // Multiplication is assigned  
            int div = 0; // Division is assigned  
            int exp = 0; // Exponent is assigned  
            int Double\_exp = 0; // Exponent for the Double Precision is assigned.   
            //decimal some = x; // NEW CODE   
            string Single\_rem = ""; // String for remainderin Single Precision is declared.   
            string Double\_rem = ""; // String for the remainder used in Double precision is declared  
            string SingleBinaryReverse = ""; // String to reverse the binary to match the exact output in Single Precision is declared   
            string DoubleBinaryReverse = ""; // String to reverse the binary to match the exact output in Double Precision is declared  
  
            //string StringReversed = "";  
  
            /\*-----------------------------------------\*/  
            /\* The next line of code will go through the procedure on  on getting the output for  
             \* SINGLE PRECISION.\*/  
            /\*----------------------------------------\*/  
  
            // If statement for when the input is greater then zero and less than one.   
            if (x > 0 && x < 1)  
            {  
                bit = 0;  
                while (x < 1)  
                {  
                    x = x \* 2;  
                    mult++;  
                }  
                mult = mult \* -1;  
                exp = 127 + mult;  
                //Console.WriteLine("exp for number between 0 and 1: " + exp);  
                Double\_exp = 1023 + mult;  
                //Console.WriteLine("exp for double precision between 0 and 1: " + Double\_exp);  
                while (exp > 0)  
                {  
                    Single\_rem = Single\_rem + exp % 2;  
                    exp = exp / 2;  
                }  
                Single\_rem = Single\_rem + "0";  
  
                while(Double\_exp > 0)  
                {  
                    Double\_rem = Double\_rem + exp % 2;  
                    Double\_exp = Double\_exp / 2;   
                }  
                Double\_rem = Double\_rem + "0";   
  
  
            }  
  
            // If statement if the user's input is greater than zero and greater than or equal to one.   
            else if (x > 0 && x >= 1)  
            {  
                bit = 0;  
                while (x >= 2)  
                {  
                    x = x / 2;  
                    div++;  
                }  
                exp = 127 + div;  
                Double\_exp = 1023 + div;  
                while (exp > 0)  
                {  
                    Single\_rem = Single\_rem + exp % 2;  
                    exp = exp / 2;  
                }  
                while (Double\_exp > 0)  
                {  
                    Double\_rem = Double\_rem + Double\_exp % 2;  
                    Double\_exp = Double\_exp / 2;  
                }  
            }  
            else  
            {  
                x = Math.Abs(x);  
                bit = 1;  
                while (x >= 2)  
                {  
                    x = x / 2;  
                    div++;  
                }  
                exp = 127 + div;  
                Double\_exp = 1023 + div;   
  
                while (exp > 0)  
                {  
                    Single\_rem = Single\_rem + exp % 2;  
                    exp = exp / 2;  
                }  
  
                while (Double\_exp > 0)  
                {  
                    Double\_rem = Double\_rem + Double\_exp % 2;  
                    Double\_exp = Double\_exp / 2;  
                }  
            }  
  
            // The string of binary for the remainders are reversed.   
            SingleBinaryReverse = StringReversed(Single\_rem);  
  
            // String for the Single Precision for the mantissa is declared  
            string S\_mantissa = "";  
  
            // Integer for the Single Precision for the mantissa is declared.   
            int S\_mantissaCount = 0;   
  
            // This part gets the decimal part of the decimal number inputted by the user.   
            double dec = (double)(x - Math.Truncate(x));  
  
            //Console.WriteLine("decimal part of inputted number: " + dec);  
  
            // Decimal part of the mantissa is multiplied by 2  
            dec = dec \* 2;  
              
            // While loop that will display 23 decimal places for the mantissa as shown on the website.    
            while (S\_mantissaCount < 23)  
            {  
                if (dec >= 1)  
                {  
                    S\_mantissa = S\_mantissa + "1";  
                    dec = dec - Math.Truncate(dec);  
                }  
                else  
                {  
                    S\_mantissa = S\_mantissa + "0";  
                }  
                dec = dec \* 2;  
                S\_mantissaCount++;  
            }  
  
  
            // Output for the Single Precision Results   
            Console.WriteLine("---------------------------");  
            Console.WriteLine("Single Precision (32 bits):");  
            Console.WriteLine("---------------------------");  
            Console.WriteLine("Bit 31 Sign Bit: " + bit);  
            Console.WriteLine("Bits 30 - 23 Exponent Field: " + SingleBinaryReverse);  
            Console.WriteLine("Bits 22 - 0 Significand: " + S\_mantissa);  
  
  
            /\*-----------------------------------------\*/  
            /\* The next line of code will go through the procedure on  on getting the output for  
             \* DOUBLE PRECISION.\*/  
            /\*----------------------------------------\*/  
  
            // The string of binary for the remainders is reversed.   
            DoubleBinaryReverse = StringReversed(Double\_rem);  
  
            // This part gets the decimal part of the decimal number inputted by the user  
            dec = (double)(x - Math.Truncate(x));  
  
            // String for the Double Precision for mantissa is declared.   
            string Double\_man = "";  
  
            // integer for the mantissa for Double Precision is declared.   
            int Double\_manCount = 0;   
  
            // Decimal part for the mantissa is multiplied by 2.   
            dec = dec \* 2;  
  
            // While loop that will display 52 decimal places as shown on the website.  
            while (Double\_manCount < 52)  
            {  
                if (dec >= 1)  
                {  
                    Double\_man = Double\_man + "1";  
                    dec = dec - Math.Truncate(dec);  
                }  
                else  
                {  
                    Double\_man = Double\_man + "0";  
                }  
                dec = dec \* 2;  
                Double\_manCount++;   
                  
            }  
  
  
            // Output for the Double Precision results.   
            Console.WriteLine("    ");  
            Console.WriteLine("---------------------------");  
            Console.WriteLine("Double Precision (64 bits):");  
            Console.WriteLine("---------------------------");  
            Console.WriteLine("Bit 63 Sign Bit: " + bit);  
            Console.WriteLine("Bits 62 - 52 Exponent Field: " + DoubleBinaryReverse);  
            Console.WriteLine("Bits 51 - 0 Significand: " + Double\_man);  
  
        }  
  
  
        // Function to reverse the stings when outputted after user makes his/her input.   
        public static string StringReversed(string rem)  
        {  
            string BinaryReverse = "";  
            for (int a = rem.Length - 1; a >= 0; a--){  
                BinaryReverse = BinaryReverse + rem[a];  
            }  
            return BinaryReverse;  
        }  
  
    }  
}