Class 4 Task Program 1: Print Difference of Two Number on LCD

```
#define _XTAL_FREQ 20000000  // Define the crystal oscillator frequency as 20MHz (for delay functions)

void Lcdinit(void);          // Function prototype for LCD initialization
void LcdCommand(uint8_t i);  // Function prototype for sending commands to the LCD
void LcdData(uint8_t i);     // Function prototype for sending data to the LCD
void LcdOutput(int16_t i);   // Modify to accept signed integer

uint8_t Equal, minus, plus;
uint16_t num1, num2;
int16_t diff;  // Use signed int to handle negative numbers

void main(void) {
    Lcdinit();  // Initialize the LCD

    num1 = 100;
    num2 = 200;
    Equal = '=';
    minus = '-';
    plus = '+';
    LcdCommand(0x80);
    LcdOutput(num1);

    LcdCommand(0x83);
    LcdData(minus);

    LcdCommand(0x84);
    LcdOutput(num2);

    LcdCommand(0x87);
    LcdData(Equal);

   diff = num1 - num2;  // Correctly stores negative values
   if(num1 > num2)
   {
       LcdCommand(0x88);
       LcdData(plus);
   }
   else
   {
       LcdCommand(0x88);
       LcdData(minus);
   }
   LcdCommand(0x89);
   LcdOutput(diff); // Correctly handles negative values

   while (1);  // Infinite loop to keep the program running
}
```

```
64  // Function to display a signed 16-bit integer on LCD
65  void LcdOutput(int16_t i) {
66      uint8_t digits[5];  // Array to store extracted digits
67      uint8_t count = 0;
68
69      if (i < 0) {
70          if (i == INT16_MIN) {
71              // Handle -32768 separately since -(-32768) overflows
72              i = 32767;  // Set to max positive value temporarily
73              digits[0] = 8;  // Store '8' separately
74              count = 1;
75          } else {
76              i = -i;  // Convert to positive for processing
77          }
78      }
79
80      if (i == 0) {
81          LcdData('0');  // Handle zero separately
82          return;
83      }
84
85      // Extract digits
86      while (i > 0) {
87          digits[count++] = i % 10;  // Store last digit
88          i /= 10;  // Remove last digit
89      }
90
91      // Print stored digits in correct order
92      for (int j = count - 1; j >= 0; j--) {
93          LcdData(0x30 + digits[j]);  // Convert to ASCII and print
94      }
95  }
```

```
98  // Function to initialize the LCD
99  void Lcdinit(void) {
100     TRISC = 0x00;  // Set PORTC as output (for control signals)
101     TRISD = 0x00;  // Set PORTD as output (for data signals)
102
103     __delay_ms(100);  // Wait for LCD to stabilize
104
105     // LCD initialization sequence as per the HD44780 LCD datasheet
106     LcdCommand(0x30);  // Send function set command (8-bit mode)
107     __delay_ms(100);   // Delay for command execution
108     LcdCommand(0x30);  // Repeat function set command
109     __delay_ms(100);
110     LcdCommand(0x30);  // Repeat function set command again
111     __delay_ms(100);
112     LcdCommand(0x38);  // Set LCD for 8-bit mode, 2-line display, 5x8 font
113     __delay_ms(100);
114     LcdCommand(0x0C);  // Turn on display, cursor off
115     __delay_ms(100);
116     LcdCommand(0x01);  // Clear the display
117     __delay_ms(100);
118 }
119
```

```
// Function to send data (characters) to the LCD
void LcdData(uint8_t i) {
    PORTC |= (0x1 << 3);  // Set RS (RC3) = 1 (indicates data mode)
    PORTD = i;            // Place data on PORTD
    PORTC |= (0x1 << 0);  // Set EN (RC0) = 1 (enable pulse start)
    __delay_ms(100);      // Small delay for command execution
    PORTC &= ~(0x1 << 0); // Set EN (RC0) = 0 (enable pulse end)
}

// Function to send commands to the LCD
void LcdCommand(uint8_t i) {
    PORTC &= ~(0x1 << 3);  // Set RS (RC3) = 0 (indicates command mode)
    PORTD = i;             // Place command on PORTD
    PORTC |= (0x1 << 0);   // Set EN (RC0) = 1 (enable pulse start)
    __delay_ms(100);       // Small delay for command execution
    PORTC &= ~(0x1 << 0);  // Set EN (RC0) = 0 (enable pulse end)
}
```