Make Up Set
Total Points: 30

Three problems must be implemented for full credit. The required (starred) problem marked in the set must be implemented by everyone. See the 2425 Grading and Submission Guide Sheets for additional grading/submission information. Partial credit will be given. (10 points each)

Note: If coding in C++, use the STL string class for problems involving strings. Do not use C style strings.

Section One- Choose two problems.

1. The hard drive for a computer recently purchased said `1EB` on the box, but when plugged in the operating system says it only had `931PB` of space. That's because hard drive marketing uses base-10 to calculate space, but computer science (and operating systems) use base-2 (and always have). So, using base-10, 1 Exabyte (`EB`) would be `1018` (`1,000,000,000,000,000,000`) bytes, but in base-2 it would be `260` (`1,152,921,504,606,846,976`) bytes. Most humans use base-10 when counting, so thus the confusion. Write a program that will take storage space given in base-10 and convert it to base-2. Use the following table for reference:

Base-10 (SI units)		Base-2 (Binary)	
8 Bits	= 1 Byte (B)	8 Bits	= 1 Byte (B)
1000 B	= 1 Kilobyte (KB)	1024 B	= 1 Kibibyte (KiB)
1000 KB	= 1 Megabyte (MB)	1024 KiB	= 1 Mebibyte (MiB)
1000 MB	= 1 Gigabyte (GB)	1024 MiB	= 1 Gibibyte (GiB)
1000 GB	= 1 Terabyte (TB)	1024 GiB	= 1 Tebibyte (TiB)
1000 TB	= 1 Petabyte (PB)	1024 TiB	= 1 Pebibyte (PiB)
1000 PB	= 1 Exabyte (EB)	1024 PiB	= 1 Exbibyte (EiB)
1000 EB	= 1 Zettabyte (ZB)	1024 EiB	= 1 Zebibyte (ZiB)

Input from the keyboard the computer hard drive size as a whole integer, a space, then a 2-letter size code reported in base-10 SI units from the table above. Output to the screen the converted size given in base-10 to base-2 units which will be reported by the operating system, rounded to two decimal places in the largest binary size you can express a whole number in. For example, do not write `1030 MiB`, write `1.01 GiB`. Finally, the program should ask if the user wants to run the program again (Check case). Refer to the sample output below.

Sample Run:

```
Enter stated hard drive size: 1 TB

Corresponding binary drive size: 931.32 GiB
```

```
Run Again (Y/N): y

Enter the stated hard drive size: 752 MB

Corresponding binary drive size: 717.16 MiB

Run Again (Y/N): N
```

Name the program: `StoreConvertXX.cpp` or `StoreConvertXX.java`, where XX are your initials.

2. Write a program to convert from decimal to negadecimal. Negadecimal, also known as base -10 (negative ten), is a non-standard positional numeral system. For example, the number 1337_{10}. in decimal, has the value one thousand three hundred thirty-seven, and can be expanded to:

$$1 \cdot 10^3 + 3 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0$$
$$(1000) + (300) + (30) + (7)$$

In negadecimal, instead of 10^n each digit would be multiplied by $(-10)^n$:

$$1 \cdot (-10)^3 + 3 \cdot (-10)^2 + 3 \cdot (-10)^1 + 7 \cdot (-10)^0$$
$$(-1000) + (300) + (-30) + (7)$$

Thus, 1337_{-10} is -723_{10} in decimal. Input from the keyboard either a positive or negative integer value. Output to the screen the represented decimal number in negadecimal. Finally, the program should ask if the user wants to run the program again (Check case). Refer to the sample output below.

Sample Run:

```
Enter decimal number: -723

Value: 1337

Run Again (Y/N): y

Enter decimal number: 21

Value: 181

Run Again (Y/N): N
```

Name the program: `NegadecimalXX.cpp` or `NegadecimalXX.java`, where XX are your initials.

3. Write a program to convert a decimal fraction to a binary fraction. For example, to convert `1/3` to binary means:

$1/3 = 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + \ldots = 0.0101\ldots_2 = 0.25 + 0.0625 + \ldots = 0.3125 + \ldots$

In the case above an exact value cannot be found with a sum of a finite number of inverse powers of two, the zeros and ones in the binary representation of `1/3` alternate forever. The point (`.`) in a binary fraction is called a radix point. Input from the keyboard two positive integer numbers separated by a space representing a rational number in `a/b` form. For example, `1 2` represents `1/2`. Output to the screen the represented rational number as a radix binary number with 6 radix places and its corresponding 6-digit decimal place value. Assume all numbers will have a digit to the left of the radix or decimal point. Finally, the program should ask if the user wants to run the program again (Check case). Refer to the sample output below.

Sample Run:

```
Enter two fractional numbers: 3 10

3/10 in binary is 0.010011 and 0.296875 in decimal.

Run Again (Y/N): y

Enter two fractional numbers: 33 25

33/25 in binary is 1.010100 and 1.312500 in decimal.

Run Again (Y/N): N
```

Name the program: `BinFractionsXX.cpp` or `BinFractionsXX.java`, where `XX` are your initials.

Required Problem- Comprehensive.

4 (**). Write an MASM program to check if a string is a palindrome. The user enters the string from the keyboard and outputs to the screen if the string is a palindrome or not. Assume proper input. Output should be user friendly.

Name the program: `StringPalindromeXX.asm`, where `XX` are your initials.

Extra Credit: Implement the following problem. See the 2425 Grading and Submission Guide Sheets for additional grading/submission information. Partial credit will be given. (10 points)

Write a MASM program that sorts a programmer-defined array. The user will enter 5 integer values from the keyboard. Do not assume the values will be entered in sorted order. Output to the screen the sorted array. Output should be user friendly.

Name the program: `ArraySortXX.asm`, where XX are your initials.