

Program 1

Due Monday Feb 11 11:59 PM

Late Programs accepted until Wednesday, Feb 12 11:59 PM (-30 points)

Decimal, Binary, and Hexadecimal

Numbers on the computer are represented using switches. If the switch is off, then the number is a zero. If the switch is on, then the number is a 1.

A number system that has only a zero digit and a one-digit is referred to as **binary**. You are used to using a **decimal** number system (some think this is because we have ten fingers).

A **hexadecimal** system, on the other hand, is one where there are 16 digits. Because we only have 10 numerals in our writing system, the numbers 10-15 are commonly represented as A thru F.

If you've ever done any web programming or messed around with Photoshop, you have probably been exposed to hexadecimal before. Colors are commonly represented with hexadecimal (hex for short) because the numerical representation is more compact than decimal and the hex representation splits up evenly into three parts (one for each of the red, green, and blue color channels). In the example below, FF represents the red channel for the color *peach*, E0 represents the green channel, and B5 represents the blue channel.



For more information on different number systems, check out the following link:

<https://www.mathsisfun.com/binary-decimal-hexadecimal.html>

One way to wrap your head around the different number systems is to line them up together in a table. The first 32 numbers are illustrated below:

Dec	Hex	Bin
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111

8	8	1000
9	9	1001
10	a	1010
11	b	1011
12	c	1100
13	d	1101
14	e	1110
15	f	1111
16	10	10000
17	11	10001
18	12	10010
19	13	10011
20	14	10100
21	15	10101
22	16	10110
23	17	10111
24	18	11000
25	19	11001
26	1a	11010
27	1b	11011
28	1c	11100
29	1d	11101
30	1e	11110
31	1f	11111
32	20	100000

You can convert from binary to decimal by using common algorithms. For example:

<https://www.wikihow.com/Convert-from-Binary-to-Decimal>

<https://www.wikihow.com/Convert-from-Decimal-to-Binary>

Similarly, there exist common algorithms for converting between hex and either decimal or binary:

<https://www.wikihow.com/Convert-Hexadecimal-to-Binary-or-Decimal>

Implementation Details

You are to create a program that is driven by an input file. An example of a file driven program will be provided below. The first line of the file will contain the number of subsequent lines in the file. The subsequent lines in the file will be four pieces of information in each line:

- 1) The number representation being used in the line (B for binary, H for hex, D for decimal)

- 2) A first number (in the representation specified in 1)
- 3) An operand, either + or *, for addition or multiplication respectively
- 4) A second number (in the representation specified in 1)

For each of the lines, you are to output the result of the calculation **in the number system specified in (1)**.

For example, if the line is **D 123 + 456**, then you will print **579** to the screen.

If the line is **B 101 + 101**, then you will print **1010** to the screen (1010).

When implementing your program, you adhere to the following guidelines:

- You are allowed to use the <string> class to read in, for example, binary, decimal, and hex values. This will probably help you parse the individual values character by character.
- You may NOT use built in C++ conversion libraries such as stol, atoi, std::hex, etc. You MUST do the conversions yourself using for loops, multiply, divide, modulo, and any helper functions to convert between characters and integers.
- You must implement at least three functions
- Because the first line of the file is a number telling you how many lines to expect in the input file, you can use a loop to iterate over the inputs.

Sample Input File

The diagram shows a sample input file with four lines. Annotations point to specific parts of the lines:

- The number of lines in the input file** points to the first line: `3`
- 'B' denotes binary values** points to the second line: `B 1011000101 + 11010001`
- 'H' denotes hexadecimal values** points to the third line: `H FFFF * 123ABC`
- 'D' denotes decimal values** points to the fourth line: `D 123 + 456`
- The operand (either addition or multiplication)** points to the `+` and `*` operators in the second and third lines respectively.

Sample Output

```
1110010110
123ABADC544
579
```

Submission Details

To canvas, you must turn in a file called **<LastName_FirstInitial>Program1.zip** containing your **program1.cpp** file, your **input text file**, your executable **program1.out** file.

Example File Driven Program

The following file driven program can be used to help you read in a file line by line.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main(){

    ifstream inFile;

    int inputNumber; // note the use of "int"
    string inputString; // note the use of "string"

    inFile.open("my_input_file.txt");

    if(!inFile.is_open()){
        cout << "Could not open file." << endl;
        return -1;
    }

    cout << "Reading first number in file..." << endl;
    inFile >> inputNumber; // notice that the first item is
                          // a number, so we read into an int
    cout << "We read the following number: " << inputNumber << endl;

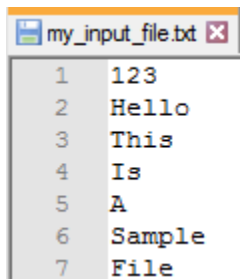
    cout << "Reading first string in file..." << endl;
    inFile >> inputString; // notice that the second item is
                          // text, so we read into a string
    cout << "We read the following text: " << inputString << endl;

    cout << "Reading second string in file..." << endl;
    inFile >> inputString; // notice that the third item is
                          // text, so we read into a string
    cout << "We read the following text: " << inputString << endl;

    // Close the file
    inFile.close();

    return 0;
}
```

The input file *my_input_file.txt* contains the following:



A screenshot of a text editor window titled "my_input_file.txt". The window displays seven lines of text, each preceded by a line number from 1 to 7. The text is as follows:

Line	Text
1	123
2	Hello
3	This
4	Is
5	A
6	Sample
7	File

The program yields the following output:

```
Reading first number in file...
We read the following number: 123
Reading first string in file...
We read the following text: Hello
Reading second string in file...
We read the following text: This
```

Example Int to Char Program

When writing this program, you will probably need to understand that there is a difference between the character '1' and the number 1. The computer will store the character '1' as a sequence of 1's and 0's in memory. It just so happens that this sequence of 1's and 0's is equivalent to 00110001, which equals **49** in base 10. As such, you will need to be aware of this fact, and write code to accommodate the conversion between characters and numbers (and vice versa). The following sample program demonstrates converting between base 10 characters and base 10 integers.

```
#include <iostream>
#include <string>
using namespace std;

int CharToNum(char value){

    // This function converts a character value into
    // an integer, assuming the character is in
    // the range of '0' to '9'

    switch(value){
        case '0':
            return 0;
        case '1':
            return 1;
        case '2':
            return 2;
        case '3':
            return 3;
        case '4':
            return 4;
        case '5':
            return 5;
        case '6':
            return 6;
        case '7':
            return 7;
        case '8':
            return 8;
        case '9':
            return 9;
    }
}

int main(){
    string myNumber = "12345";
```

```

        // loop over myNumber until we hit myNumber.length()
        // printing out the individual numbers line by line
        for(int i = 0; i < myNumber.length(); i++){
            int x = myNumber[i]; // note that we are NOT doing a proper conversion
from char to int
            cout << x << endl;
        }
        cout << endl;

        for(int i = 0; i < myNumber.length(); i++){
            int x = CharToNum(myNumber[i]); // now we are using a helper function to
do the appropriate conversion
            cout << x << endl;
        }
        cout << endl;
        return 0;
}

```

Output:



Example Use of Modulo and pow

The modulo operator (%) and the pow function are two handy tools to use when manipulating numbers.

Modulo (or mod) returns the remainder of a division.

For example:

$3 / 2 = 1.5$, or, 1 with a remainder of 1.

$4 / 2 = 2.0$, or 2 with a remainder of 0.

$5 / 2 = 2.5$, or 2 with a remainder of 1.

And so on...

The pow function is simply the way to perform an exponential operation, x^y , in c++.

```

#include <iostream>
#include <math.h> // contains the pow function
using namespace std;

int main(){
    int myValue = 15;

    cout << "myValue % 2: " << myValue % 2 << endl;
    cout << "myValue % 10: " << myValue % 10 << endl;
    cout << "myValue % 16: " << myValue % 16 << endl;

    double result = pow(10,2); // note that pow returns a DOUBLE, not an INT
    cout << "10 ^ 2: " << result << endl;
    return 0;
}

```

Output:

```

myValue % 2: 1
myValue % 10: 5
myValue % 16: 15
10 ^ 2: 100

```

Example Use of long long int

Different data types use a different amount of 1's and 0's to represent the number.

For example, on a 32 bit system, an integer is represented with 32 bits. This means that the maximum value for an integer is $2^{31}-1 = 2,147,483,647$ (the first bit is used to control if the integer is positive or negative), so it's $2^{31}-1$ and not $2^{32}-1$.

The effect of this limitation is that if an integer gets too large, the processor will flip the first bit and make the number appear negative.

A way to avoid this problem is to use a data type that has a greater amount of 1's and 0's. For example, the data type **long long int** on the same 32-bit system will have a max value of $2^{63}-1$.

See below for an example of creating a number too large to be represented by an int, but is easily represented by a **long long int**.

```

#include <iostream>
using namespace std;

int main(){
    // The largest value for INT is 2,147,483,647
    int num1 = 1000000000; // 1,000,000,000
    int num2 = 1000000000; // 1,000,000,000
    int num3 = num1 * num2; // Should be 1,000,000,000,000,000

    cout << "Num3 as INT: " << num3 << endl;

    long long int num4 = 1000000000; // 1,000,000,000
    long long int num5 = 1000000000; // 1,000,000,000
    long long int num6 = num4 * num5; // Should be 1,000,000,000,000,000
}

```

```

        cout << "Num6 as LONG LONG INT: " << num6 << endl;
        return 0;
}

```

Output:

```

Num3 as INT: -1486618624
Num6 as LONG LONG INT: 1000000000000000000

```

Reversing a String

When building this program, you may find it useful to reverse strings. There is a *reverse* algorithm located in the *algorithm* library.

The program below shows you how you can reverse a string, as well as append characters to a string and then do a reversal.

```

#include <iostream>
#include <string>
#include <algorithm> // contains the 'reverse' algorithm
using namespace std;

int main(){
    string mySentence = "This is a sentence.";
    cout << mySentence << endl;

    reverse(mySentence.begin(), mySentence.end()); // reverse the string

    cout << mySentence << endl;
    cout << endl;

    string myNumberAsString = "";

    // append characters to string
    myNumberAsString += '1';
    myNumberAsString += '2';
    myNumberAsString += '3'; // the string is now "123"

    cout << myNumberAsString << endl;

    reverse(myNumberAsString.begin(), myNumberAsString.end()); // reverse

    cout << myNumberAsString << endl;
    cout << endl;

    return 0;
}

```

Output:

```

This is a sentence.
.ecnetnes a si sihT

123
321

```