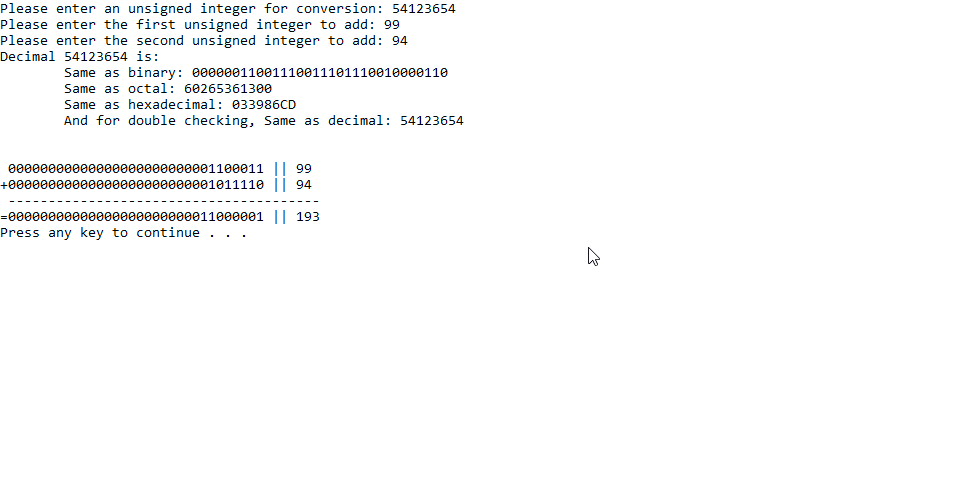
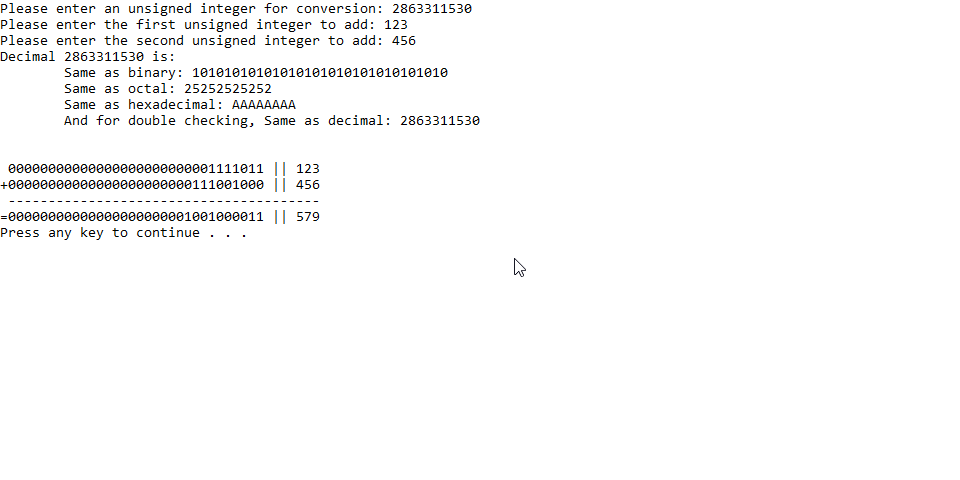
**Computer Science 2253 Lab Experience Two**



/\*

\* johnathanLeeLab2.cpp

\* Author: Johnathan Lee

\* Date: 9/7/17

\*

\* Collection of functions for conversion of unsigned integers to binary,

\* conversion of binary strings to octal/hex/dec, and addition of binary numbers.

\*/

#include <iostream>

#include <string>

#include <assert.h>

#include <iomanip>

using namespace std;

string unsignedToBinary(unsigned int num);

string arrayToString(int ar[32]);

void displayBinary(string input);

void displayOctal(string input);

void displayHex(string input);

void displayDec(string input);

void addBinary(string strNum1, string strNum2, int result[32]);

void runTests(unsigned int num, unsigned int addNum1, unsigned int addNum2);

int main() {

bool gotNum = true;

unsigned int convNum, addNum1, addNum2;

cout << "Please enter an unsigned integer for conversion: ";

cin >> convNum;

cout << "Please enter the first unsigned integer to add: ";

cin >> addNum1;

cout << "Please enter the second unsigned integer to add: ";

cin >> addNum2;

runTests(convNum, addNum1, addNum2);

return 0;

}

// unsignedToBinary

// Converts an unsigned integer to its 32-bit binary representation.

// Input: An unsigned binary number.

// Output: A string containing the unsigned 32-bit representation.

// (with leading zeroes)

string unsignedToBinary(unsigned int num) {

int remainder;

string finalNum = "";

while (num != 0) {

remainder = num % 2; //cout << "Remainder: " << remainder << '\n';

num /= 2; //cout << "Dividend: " << dividend << '\n';

finalNum = to\_string(remainder) + finalNum;

}

while (finalNum.length() != 32) {

finalNum = "0" + finalNum;

}

// assert(finalNum.size() == 32); Left in from debugging

return finalNum;

}

// arrayToString

// Helper function to turn an array of single digit integers to a string.

// Input: An array of size=32 filled with integers.

// Output: Returns a string with the character counterparts of each int in ar.

string arrayToString(int ar[32]) {

string result;

for (int i = 0; i < 32; i++)

result += to\_string(ar[i]);

return result;

}

// displayBinary

// Displays a binary number-string. No newline at end.

// Input: A string containing a binary number.

// Output: Prints the input to the console, since it's the same as needed.

void displayBinary(string input) {

cout << input;

}

// displayOctal

// Turns a 32-bit bin number into its octal counterpart. No newline at end.

// Input: A string containing the number in binary. (where length == 32).

// Output: The octal representation of the binary input string is printed.

void displayOctal(string input) {

// assert(input.length() == 32); Left in from debugging

string parts[11]; // 8=2^3 -> 32/3 = 10.667 -> 10+1 = 11

string finalNum = "";

parts[0] = input.substr(0, 2); // 32 % 3 == 2, so take the leading 2 off seperately.

parts[0] = '0' + parts[0]; // Makes things easier later since they're all 3 long.

// 35 instead of 32 because we go up 3.

for (int i = 2, cnt = 1; i + 3 < 35; i += 3, cnt++)

parts[cnt] = input.substr(i, 3);

for (int i = 0; i < 11; i++) {

string& part = parts[i];

// '1'-'0' = 49-48 = 1, and '0'-'0' = 0. (Convert ascii to 1 or 0).

// Then, follow normal conversion of b^n-1 \* d

// where b = base, n = place, and d = either 0 or 1 (digit).

int digit = (part[0] - '0') \* 4

+ (part[1] - '0') \* 2

+ (part[2] - '0') \* 1;

finalNum = to\_string(digit) + finalNum;

}

cout << finalNum;

}

// displayHex

// Turns a 32-bit bin number into its hexadecimal counterpart. No newline at end.

// Input: A string containing the number in binary. (where length == 32).

// Output: The hexidecimal representation of the binary input string is printed.

void displayHex(string input) {

string parts[8], // 16=2^4 -> 32 / 4 = 8 -> We get 8 hex digits.

finalNum = "";

for (int i = 0, cnt = 0; i + 4 < 33; i += 4, cnt++)

parts[cnt] = input.substr(i, 4);

for (int i = 0; i < 8; i++) {

string& part = parts[i];

int digit = (part[0] - '0') \* 8

+ (part[1] - '0') \* 4

+ (part[2] - '0') \* 2

+ (part[3] - '0') \* 1;

if (digit < 10) { // We don't need to change to A-F

cout << digit;

} else {

// 11-10 = 1 -> 'A' + 1 = 'B', 'A' + 2 = 'C', and so on.

int offset = digit - 10;

finalNum = static\_cast<char>('A' + offset) + finalNum;

}

}

cout << finalNum;

}

// displayDec

// Turns a 32-bit bin number into its base-10 counterpart. No newline at end.

// Input: A string containing the number in binary. (where length == 32).

// Output: The decimal representation of the binary input string is printed.

void displayDec(string input) {

unsigned int finalNum = 0;

for (int i = 0; i < 32; i++) {

bool binDigit = (input[i] - '0');

finalNum += static\_cast<unsigned int>(pow(2, 31 - i)) \* binDigit;

}

cout << finalNum;

}

// addBinary

// Adds 2 binary numbers and prints the operations done, storing the result.

// Input: A string for each number to be added, an integer array of size=32

// to store the result in.

// Output: A prettified version of the operations carried out is printed to

// console and the result of the operations (in binary) is stored

// in result.

void addBinary(string strNum1, string strNum2, int result[32]) {

int num1[32] = { 0 }, num2[32] = { 0 };

int carry = 0;

// Convert the strings to int arrays.

for (int i = 0; i < 32; i++)

num1[i] = static\_cast<int>(strNum1[i] - '0');

for (int i = 0; i < 32; i++)

num2[i] = static\_cast<int>(strNum2[i] - '0');

for (int i = 31; i >= 0; i--) {

int curRes = num1[i] + num2[i] + carry;

if (curRes >= 2) { // 1+1 or 1+1+(carried) 1

carry = 1;

// bin 11 carries a 1 and keeps the 1 in the 1's place.

result[i] = (curRes == 3 ? 1 : 0);

} else { // 1+0 or 0+0+(carried) 1

carry = 0;

result[i] = curRes;

}

}

// Pretty print in the following format:

// 00000000000000000000000000000000 || 0000000000

//+00000000000000000000000000000000 || 0000000000

//-----------------------------------------------

//=00000000000000000000000000000000 || 0000000000

cout << " " << arrayToString(num1) << " || ";

displayDec(arrayToString(num1));

cout << "\n+" << arrayToString(num2) << " || ";

displayDec(arrayToString(num2));

cout << "\n ---------------------------------------\n="

<< arrayToString(result) << " || ";

displayDec(arrayToString(result));

cout << endl;

}

void runTests(unsigned int num, unsigned int addNum1, unsigned int addNum2) {

int result[32] = { 0 };

string binValue = unsignedToBinary(num);

cout << "Decimal " << num << " is: ";

cout << "\n\tSame as binary: ";

displayBinary(binValue);

cout << "\n\tSame as octal: ";

displayOctal(binValue);

cout << "\n\tSame as hexadecimal: ";

displayHex(binValue);

cout << "\n\tAnd for double checking, Same as decimal: ";

displayDec(binValue);

cout << "\n\n\n";

addBinary(unsignedToBinary(addNum1), unsignedToBinary(addNum2), result);

}