Tatyana Vlaskin
Assignment 2 Report

1. Convert (feel free to just use paper, though I use a text editor for this sometimes) the following numbers from decimal to binary: (**turn this in as a part of the design document later)**

a. 3
b. 7
c. 10
d. 50
e. 94
f. 192

To convert decimals to binary Ill be using the following steps.

1. Create table whose leftmost column is the greatest power of 2 less than the number that needs to be converted.
2. Compare the number that needs to be converted to the value at the leftmost column
3. If the leftmost column is less than the number that you need to convert, put 1 in the column. Sum of the values of all the columns with 1 is defined as running total.
4. Go to the next column, add this value to the value of the leftmost column (this is the running total at this point)
5. If the running total is less than the number that needs to be converted, place 0
6. If the running total is greater than the running total place 0
7. Go to the next column
8. Repeat steps 4-7 until all columns are filled
9. At the end your running total should add up to the converted number

a. 3

$2^{\wedge 1} = 2$	$2^{\wedge 0} = 1$	
1		2<3 yes
	1	2+1 = 3 yes, done

Binary number: 11

b. **7**

$2^{\wedge 2} = 4$	$2^{\wedge 1} = 2$	$2^{\wedge 0} = 1$	
1			4 < 7 yes
	1		4 +2 < 7 yes
		1	

Binary number: 111

c. 10

$2^{\wedge}3 = 8$	$2^{\wedge}2 = 4$	$2^{\wedge}1 = 2$	$2^{\wedge}0 = 1$	
1				8 < 10 yes
	0			8+ 4 < 10 no
		1		8 + 2 = 10 yes, done
			0	8+ 2+ 1 <10, no

Binary number: 1010

d. 50

$2^{\wedge}5 = 32$	$2^{\wedge}4 = 16$	$2^{\wedge}3 = 8$	$2^{\wedge}2 = 4$	$2^{\wedge}1 = 2$	$2^{\wedge}0 = 1$	
1						32 < 50 yes
	1					32 + 16 <50 yes
		0				32 +16 +8 < 50 no
			0			32 +16 +4 <50 no
				1		32 +16 +2 = 50 done
					0	Done

Binary number: 110010

e. 94

$2^{\wedge}6 = 64$	$2^{\wedge}5 = 32$	$2^{\wedge}4 = 16$	$2^{\wedge}3 = 8$	$2^{\wedge}2 = 4$	$2^{\wedge}1 = 2$	$2^{\wedge}0 = 1$	
1							64 < 94 yes
	0						64 + 32 <94 no
		1					64+16 < 94 yes
			1				64+16+8<94 yes
				1			64+16+8+4 <94 yes
					1		64 +16+8+4+2 = 94 done
						0	Done

Binary number: 1011110

f. 192

$2^{\wedge}7 = 128$	$2^{\wedge}6 = 64$	$2^{\wedge}5 = 32$	$2^{\wedge}4 = 16$	$2^{\wedge}3 = 8$	$2^{\wedge}2 = 4$	$2^{\wedge}1 = 2$	$2^{\wedge}0 = 1$	
1								128 < 192 yes
	1							128 +64 = 192 done
		0						Done
			0					Done
				0				Done
					0			Done
						0		Done
							0	done

Binary number: 11000000

2. Convert the following numbers from binary to decimal: **(turn this in as a part of the design document later)**

a. 10
b. 1110
c. 111010
d. 11100011

To convert binary to decimal, Ill do the follow steps:

1. Make a table with enough columns for the binary number
2. Fill in the binary number in the table
3. Add all value in columns with number 1

a. 10

$2^{1} = 2$	$2^{0} = 1$
1	0

Total: 2 = 2

b. 1110

$2^{3} = 8$	$2^{2} = 4$	$2^{1} = 2$	$2^{0} = 1$
1	1	1	0

Total: 8+4+2 =14

c. 111010

$2^{5} = 32$	$2^{4} = 16$	$2^{3} = 8$	$2^{2} = 4$	$2^{1} = 2$	$2^{0} = 1$
1	1	1	0	1	0

Total: 32+16+8+2=58

d. 11100011

$2^{7} = 128$	$2^{6} = 64$	$2^{5} = 32$	$2^{4} = 16$	$2^{3} = 8$	$2^{2} = 4$	$2^{1} = 2$	$2^{0} = 1$
1	1	1	0	0	0	1	1

Total: 128+64+32+2+1 = 227

3. Convert each of the decimal numbers from problem 1 above into two's compliment representation numbers with the **minimum number of bits** possible. This should not be difficult, and is more to get you thinking about the sign bit than about the rest of the number (which behaves very much like binary).**(turn this in as a part of the design document later**

To convert a decimal number into two's complement representation numbers, I'll follow these steps.

i. are done Convert the number to binary. (THIS WAS ALREADY DONE IN PROBLEM 1)
ii. If the number is positive add 0 in front of the binary and you

a. 3 in binary is 11

Using a 3 bit representation 3 is <u>011</u> in binary using 2s compliment notation.

b. 7 in binary is 111

Using a 4 bit representation 7 is <u>0111</u> in binary using 2s compliment notation.

c. **10 in binary is 1010**

Using a 5 bit representation 10 is <u>01010</u> in binary using 2s compliment notation.

d. **50 in binary is 110010**

Using a 7 bit representation 50 is <u>0110010</u> in binary using 2s compliment notation.

e. **94 in binary is 1011110**

Using an 8 bit representation 94 is <u>01011110</u> in binary using 2s compliment notation.

f. **192 in binary is 11000000**

Using an 9 bit representation 192 is <u>011000000</u> in binary using 2s compliment notation

4. Now convert them each into an 8-bit two's compliment representation, but have each one be negative what it originally was (so write -3 in 8-bit two's complement, -7, -10, -50, or whatever the numbers were in problem 1 but now negative, written out in two's complement).(turn this in as a part of the design document later

   To convert a negative number into 8-bit 2s complement, I need to do the following steps:

   i. Convert positive number into the binary and find 2s complement of the positive # (THIS WAS ALREADY DONE IN PROBLEMs 2 and 3)
   ii. Express 2s compliment in 8 bits
   iii. Find the complement of the positive 2s complement by inverting 0s and 1s
   iv. Add 1 to the complement

a. 3 in binary is 11

Using an 8-bit representation +3 in 2s complement is:	00000011
Complement	11111100
Add 1	00000001
-3	**11111101**

b. 7 in binary is 111

Using an 8-bit representation +7 in	00000111

2s complement is:	
Complement	11111000
Add 1	00000001
-7	**11111001**

c. **10 in binary is 1010**

Using an 8-bit representation +10 in 2s complement is:	00001010
Complement	11110101
Add 1	00000001
-10	**11110110**

d. **50 in binary is 110010**

Using an 8-bit representation +50 in 2s complement is:	00110010
Complement	11001101
Add 1	00000001
-50	**11001110**

e. **94 in binary is 1011110**

Using an 8-bit representation +94 in 2s complement is:	01011110
Complement	10100001
Add 1	00000001
-94	**10100010**

f. **192 in binary is 11000000**

To represent 192 in binary using 2s compliment notation, we need at least 9 bits: 011000000 (see problem 3). The requirement of this problem is to use 8-bits. For 192 this will result in overflow; thus -192 cannot be expressed into an 8-bit 2s complement because there is an overflow.

5. What happens if you add two very large positive 8-bit two's complement numbers together (say 01100110 and 01011100)?

Number 1	0	1	1	0	0	1	1	0
Number 2	0	1	0	1	1	1	0	0
sum	1	1	0	0	0	0	1	0

The value of the bit carried into the leading digit (the most significant bit) has changed from 0 to 1. This is an overflow and this result is incorrect. Leading 1 indicates that result is negative, which does not make sense since we added two positive 2s complement numbers.

6. What happens if you add two very large negative 8-bit two's complement number together (say 10100001 and 10101010)?

Number 1	1	0	1	0	0	0	0	1
Number 2	1	0	1	0	1	0	1	0
sum	0	1	0	0	1	0	1	1

Similar to the previous problem, the value of the bit carried into the leading digit (the most significant bit) has changed. But in this case it changed from 1 to 0. At the same time, there is 1 that needs to be carried to the 9^{th} bit; however we have only 8 bits. This is another example of an overflow and the result is incorrect.
Leading 0 indicates that result is positive, which does not make sense since we added two negative 2s complement numbers.

8.
Meeting Room Programming Assignment: JUST READ ON THE DISCUSSION BOARD, WE DO NOT NEED TO DO THIS. TOO LATE

Understanding:

For this project I need to write a program that determines if a meeting can be held in a room with specified maximum room capacity.
The program requires the following:

1. Asks the user to indicate room capacity (variable #1) (OUTPUT)
2. Takes the input from user and stores it as a variable #1 (INPUT)
3. Asks the user to indicate how many people are at the meeting (OUTPUT) (variable #2)
4. Takes the input from the user and stores it as a variable #2 (INPUT)
5. Compares room capacity (variable #1) and # of people at the meeting (variable #2).
    a. If # of people exceeds the room capacity, program lets the user know that they cannot have a meeting unless certain # of people will leave the room (variable #3) (OUTPUT in if statement)
    b. If # of people is less or equal to room capacity, program lets the user know that meeting can be held and indicates if any additional people are allowed to come to the meeting. It also specifies how many additional people can come to the meeting (variable 3) (OUPTUP possibly in else statement).
6. I think it's a good idea to let the user know if the

Design:

1. Declaration of variables:
   int people, int capacity
2. cout statement asking user to enter room **capacity**
3. cin statement to take in user input for **capacity**
4. cout statement asking user for # of **people** at the meeting
5. cin statement to take in users input for # of **people** at the meeting
6. If capacity equals to or less than # of **people** at the meeting
   cout statement indicating that meeting is allowed
   declaration of variable int difference = capacity – people
   cout statement of difference, which will indicate how many people can still joint the meeting

   else if capacity is greater than # of **people** at the meeting

   cout statement indicating that its not legal to have a meeting

   cout statement of absolute value of different to let the user know

   how many people have to leave in order to have a legal meeting.

   a. return 0 to end the program

Reflection:

I've decided to enter a character instead of number for number of coupons and program gave really strange output. I got 213056716 for the number of candies, 2 for the number and gums and 2 for coupons left. I am guessing when program expected int type input, but wrong type was entered and program started acting weird not knowing what to do with incorrect type of input:
When I wrote my design it did not come to my mind that I need to include a statement that lets the user know when incorrect type of input is entered and either asks to make a new entry or lets the user know that program stopped working due to invalid entry. This was not covered in the class, so I had to do additional reading for this. I came across the following blog:
http://www.cplusplus.com/forum/beginner/49969/
I've decided to add the following lines to my code:

```
if (!cin.good())
{
    cout << "Your entry is invalid. Please make a valid entry. \n";
```

```
        cin.clear ();
        cin.sync ();
{
```
This fixed the problem.

PROBLEM 9: JUST READ ON THE DISCUSSION BOARD WE DO NOT NEED TO DO THIS. TOO LATE

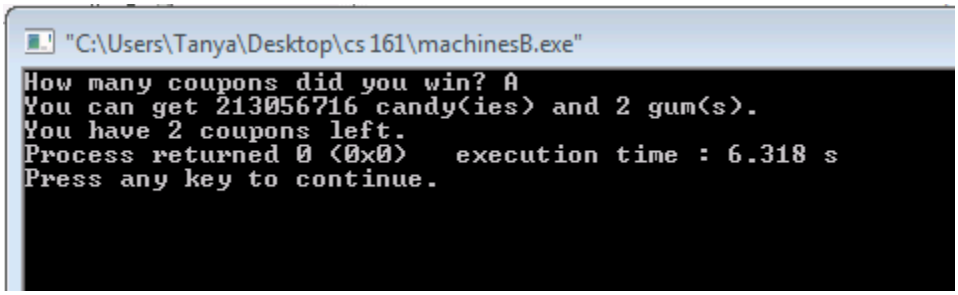
FOR GUESSING GAME

Reflection:

When I tested the program, I really had a hard time figuring out how to get the correct output for the number of coupons left. I used the following expression to find the number of coupons left: couponsLeft = coupons - (candy*10 + gum*3). Initially I put it after 2 while loops. When I tried to run the program by entering 3 for number of coupons to be redeemed, I kept getting - 3 for coupons left. After I looked at the code, I've realized what the problem was. Each time loop is executed, the number of coupons is reduced:

```
//loop is entered when there are < 10, but >= 3 coupons
while (coupons < 10 && coupons >= 3)
{ coupons = (coupons - 3);   //removes coupons for each gum
    gum++;//    counts gums
}
     couponsLeft = coupons - (candy*10 + gum*3)
```

So, if initially there were 3 coupons, after this loop, # of coupons is equal to 0.
After the loop was executed for the last time, coupons variable represented how many coupons were left. Thus, couponsLeft = coupons - (candy*10 + gum*3) was giving me incorrect output.
After I've removed couponsLeft = coupons - (candy*10 + gum*3) statement from my program, it was giving me correct output. I decided that I was done with this program. And then all of sudden, I've decided to enter a character instead of number for number of coupons and program gave really strange output. I got 213056716 for the number of candies, 2 for the number and gums and 2 for coupons left. I am guessing when program expected int type input, but wrong type was entered and program started acting weird not knowing what to do with incorrect type of input:



When I wrote my design it did not come to my mind that I need to include a statement that lets the user know when incorrect type of input is entered and either asks to make a new entry or lets the user know that program stopped working due to invalid entry. This was not covered in the class, so I had to do additional reading for this. I came across the following blog: http://www.cplusplus.com/forum/beginner/49969/

I've decided to add the following lines to my code:

```
if (!cin.good())
{
      cout << "Your entry is invalid. Please make a valid entry. \n";
      cin.clear ();
      cin.sync ();
{
cout << "How many coupons did you win? ";
cin >> coupons;
```

After this, I've tested the code by taking random entries and it seems that the problem is solved.

PROBLEM #12 PER DISCUSSION BOAD, WE ONLY NEED REPORT FOR THIS GAME

Understanding:

1. Declaration of the variables – still trying out to figure out what needs to be declared
2. User number 1 is asked to for a number? Indicates acceptable range for numbers (OUTPUT STATEMENT)
3. User number 1 enters a number (INPUT STATEMENT)
4. If the number is outside indicated range, user 1 is asked to reenter the number
5. We do NOT need to worry if user 1 enters bad type input (word, string, etc.) per instructions. Basically, we are making an assumption that user will NEVER make bad type input. Thus, in the testing of the program, I do NOT need to check how the program will behave when bad data type is entered.
6. If the number entered falls in the range specified by the program, the number is accepted and stored as a variable #1

7. User number 2 is asked to guess a number. Range of possible numbers is indicated to the 2 user. Lets the 2 user know how many guesses they have (OUTPUT STATEMENT).
8. User number 2 enters their guess (INPUT) , the number is accepted and stored as a variable #2
9. Variable #2 is compared to Variable #1
    a. If Variable #2 equals Variable #2, user #2 notified that they are correct and displays congratulations message
    b. If Variable #2 is higher Variable # 1, lets the user know that the guess is too high and asks the user to make a new entry.
    c. If Variable #2 is lower than Variable #1, lets the user know that the guess is too low and asks the user to make a new entry
10. With each guess the user makes, there is a change to the "range of numbers that reflect the effect that the user's new guess had on the range of valid numbers." Lets say original range was 0-10. User number #1 entered 3, while user # 2 guessed 7. Guess is too high, so range will be changed to 0-6.
11. When the user #2 runs out of guess, notifies the user that they ran out of guesses and what the secret number was and how close their closes guess was (displays the difference between number and closest guess).
12. At the end of the game, asks the user #2 whether they want to play again.

Design:

Bool operator: try or not try again

while (try again) continue playing
{
define variables for:
1. User 1 input
2. User 2 input
3. Attempts = 5 (set by programmer)
4. Define range – this will be defined by the programmer
   Ill define my range from 0 to 100

FOR THE USER 1
Do loop
{
Asks a user to enter a number in a certain range and continue the loop until value falls in the correct range.

if(numberUser1 > max)
{ lets the user know that entered number was too large

And asks to reenter a number
}
else if(numberUser1 < min)
{lets the user know that the number is small,
Asks the user to reenter a number}
}
} this loop will continue when:
while (numberUser1 > max || numberUser1 < min)

this part is for the user 2 to guess the number that was entered
Asks a user to guess a number
Indicates the range of the umbers
Lets the user know how many attempts they have

Declaration of bool correctGuess = false
(until this loop is changed to true when the user guesses the number this Loop will be executed
while (correctGuess == false)
{
LOOP IS ENTERD IF THE USER GUESSED THE NUMBER
if (numberUser2 == numberUser1 && guesses >= 0)
{
Congratulation message
Changing of bool to true
This will make the user directly to the question if they want to play one more time

}
If the number is not gueesed
Else (this will count how many guesses are left)
{
}
If the user entered a number that is higher than max
if(numberUser2 > max && guesses > 0)
{
Invalid entry message
Asks a user to reenter number
Range is not changed because entry is invalid
Indicates number of guesses left
}

```
            If the user enters a number that is too high and it is in acceptable
            range
             else if (numberUser2 > numberUser1 && guesses > 0)
             {
                Lets the user know that number is too high
               Asks a user to reenter number
               Changes a range and lets the user know that is the new range
               Indicates number of guesses left

             }

             //IF THE USER ENTERS NUMBER THAT IS HIGHER THAN
MIN
             // and there are still guesses left
             else if(numberUser2 < min && guesses > 0)
             {
                 cout << "INVALID ENTRY! "; // lets the user know that invalid
entry was made
                 //lets the user know what was the problem with the number
entered
                 cout<< numberUser2<<" is lower than the allowed range. Try
again.\n "
                 << "You have " << guesses  <<" guesses left. \n" ;
                 //range is not changed when invalid entry is made
                 cout << "Please enter a number from " <<min <<" to " << max
<<". ";
                 cout << "Enter a number: ";//asks user # to enter the number
                 cin  >> numberUser2; // alows the user to enter a number
                 cout << "\n"; //new line
             }

            If the user enters a number that is too low and it is in acceptable
            range
             else if (numberUser2 < numberUser1 && guesses > 0)
             {
                Lets the user know that number is too low
               Asks a user to reenter number
               Changes a range and lets the user know that is the new range
               Indicates number of guesses left

             }

               If there are no more guesses left
```

{

Lets the user know that they lost the game
Lets the user know what was the correct number
LETS THE USER KNOW HOW CLOSE THEY WERE
(at this point I have no idea how to do this)

} correctGuess = true; // even though, user did not indicate correct number, bool is changed to true to make sure that we do not enter loop any more

}

declaration of the variable for a user to decide if use wants to play one more time
char can be either ‘y’ or ‘n’
Asks the user if they want to play a game one more time

executed only if user does not want to play againneed a n or no answer
if (decision == 'n' || decision == 'N')
{
playAgain = false; // change the bool to false and play again loop is not entered

}
}
END

TESTING:

testing	Input (tested conditions)	Expected output	Pass/Fail
User #1 input	Number > max	Invalid entry, asks to reenter #	pass
User # 1 input	Number <min	Invalid entry, asks to reenter #	pass
User #1 input	Number in the range	Asks user # 2 to guess a number	Pass
User # 2	Guess a number	Congratulation message And asks a user if they want to	pass

		play one more time	
User #2	Guess a number and say yes for replay	Game starts over	Pass
User #2	Guess a number and say no for replay	Exit game	Pass
User #	Enters number that is too low, and in the range	Asks to reenter a number, adjusts min and counts guesses	Pass
Use 2	Enters a number that is too high	Asks to reenter a number, adjusts max and counts guesses	Pass
User 2	Enters a number that is smaller than min	Gives invalid message range, does not change a range, but counts guesses	Pass
User 2	Enters a number that is smaller than max	Gives invalid message range, does not change a range, but counts guesses	Pass
User 2	Runs out of guesses	Lets the use know that they ran out of guess	Pass
User 2	Runs out of guesses	Lets the user know what is the true value	Pass
User 2	Runs out of guesses	Asks the user if they want to play again. If they say yes, game starts over	Pass
User 2	Runs out of guesses	Asks the user if they want to play again. If they say no, exits the screen	Pass

User 2	Runs out of guesses	Lets the users know how close, they were.	I HAVE NO IDEA HOW TO DO THIS. IF I FIGURE IT OUT, ILL MAKE A NOTE IN THE REFLECTION (PASS – I THINK IT WORKS)

Reflection:
I had a hard time with this program. First of all I've used bool operator to determine if the number was guessed correctly. At the beginning, I've assigned it to bool correctGuess = false and once the user guessed the number, I was planning to switch it to true. The problem was that instead of = operator I've used == operator and I've encountered infinite look. It took me at least an hour to figure out what was the problem.

```
 while (correctGuess == false && guesses > 0)
{
if (numberUser2 == numberUser1) // if the number was quessted.
{
   cout << "Congratulations!!! You guesed the number. Game is
over!";
correctGuess == true; (THIS CAUSED A LOT OF PROBLEMS).
}
```

Another problem that I've encountered is that when number of guesses was equal to 0, a usre was asked to make a guess anyways. The braching loop: while (correctGuess == false && guesses > 0), was executed for some reason even when guesses = 0. I have no idea why that was happenidng and it does not make any sense to me. To fix the problem, I've added bool condition to each loop inside the while (correctGuess == false && guesses > 0); such as:
if (numberUser2 > numberUser1 && guesses > 0) and change while loop from while (correctGuess == false && guesses > 0) to while (correctGuess == false); and itit fixed the problem.

At this point I realized that I forgot to write a cope to let the user know that they are out of guesses and what was the correct number and how close

their closes guess was. I had to idea what to do to display how close they were, so I decided to postpose writing that that part. As fulfull the rest of the requirements, I've decided to do something like this, which kept giving me an error:

```
else if (numberUser1 != numberUser2 && guesses = 0)
{
   cout << " You do not have any guesses left. \n"
   << " The secret number was " << numberUser1
  << "Game is over." <<   "Your number was
```

The error was: guessingGame2.c|91|error: non-lvalue in assignment|
I've decided to replace: else if (numberUser1 != numberUser2 && guesses = 0) with "else" and program compliped, but when I ran the program , I realized that this loop was infinite. I've decided to add add the following lineinside the loop: correctGuess = true and it fixed the problem.
My next step was to figure out how to let the user know how close their closest guess was. This part I completely ignored in design section because I had no idea how to approach it and I was about to completely gave up on this. But I kept thinking and all of a sudden I remembered that we kept changing the range of numbers to reflect the effect that the users newest guess had on the range of valid value. So basically when the user is out of guesses our new range is from (users lowest guess+1) = min to (users highest guess -1) = max. What we can do next is find the difference between numberUser 1 and users lowers guess and numberUser1 and users highest guess and whichever gives the lowest absolute value (fabs (difference)), it the closest guess. I decided to give it a try.

On a lower side: numberUser1 – (min -1) = differenceMin

On the upper side: numberUser1 – (max +1) = differenceMax

If (differenceMin < differenceMax)

{ cout << "Your closes guess was" << (numberUser2 -1);

}

Else

{ cout << "Your closes guess was" << (numberUser2 +1);

}

It looked very promising, until user #2 tried to enter numbers that exceeded the max or were smaller than min of the range. To fix the problem, I've decided to add 2 additional loops to let the user #2 know that the number entered is invalid. If the user #2 entered a number higher/lower than the max/min of the range, it was counted as an incorrect guess. This did not work, I spent hour figure out what was going on and finally came up with this code that seems to work. See comments in the code. The only time it does not work is user #2 keeps entering negative invalid numbers. I wish I had more time to work on this to fix the problem. Overall, after this assignment I definitely know how to use loops.