William George
Report for Assignment 2
CS 161 Fall 2014
Week 2 October 8, 2014

Understanding:

For this week's exercises we are working on converting binary to decimal and vice versa. Also I worked on two's complement. The purpose of the two's complement is to see how the sign bit works and to understand the behavior of binary when it is added and subtracted. Also there was some limited information on overflow and the importance of bit space. Some numbers no matter whether they are positive or negative require different amounts of bit space than others.

This week's programing assignments had the purpose of helping develop many skills we will need as our programs get more complex:

- The fireLaw program was for practicing the "if" conditional statement to show how we can get commands to execute only when certain conditions are met. Also the program will help with understanding variable usage throughout the program. For my program I will use the int and the const int variable types.

- The first arcade program was used to practice using some of the operators, specifically the division and remainder operators.

- The second arcade program, arcade2's purpose was to practice looping and practicing choosing the correct loop to do the commands you need the program to do. There are several different loops that can be used for different scenarios depending on whether you want to pretest a condition or post-test and if you know how many times the program will run through the loop i.e. you want the program to run the loop once or to skip it if the looping condition is already met. Also we will practice nested loops and if statements here.

- Finally, the programming project for the week will address both loops, if statements, variable declarations and general programming logic.

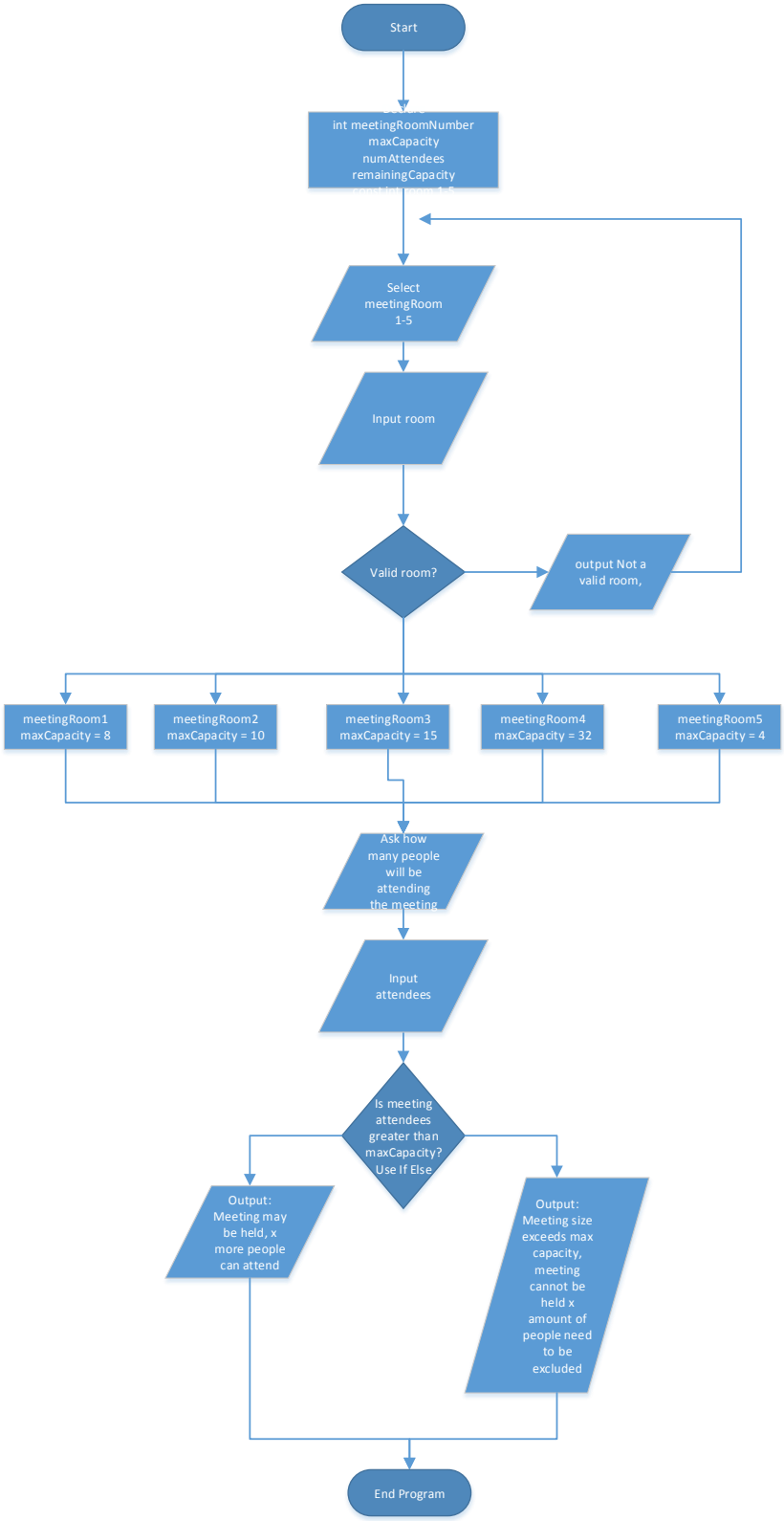
Design

1. Program 1: fireLaw.cpp

   Pseudocode

```
START int main()
DECLARE variables
        const int roomOne = 8, roomTwo = 10, roomThree = 15, roomFour = 32, roomFive =4;
        int meetingRoomNumber, maxCapacity, numAttendees, remainingCapacity;

OUTPUT: “Please select meeting room 1,2,3,4 or 5”;
INPUT: meetingRoomNumber;

WHILE meetingRoomNumber is not 1-5
        OUTPUT: “That is not a valid meeting room number please select room 1,2,3,4,5”;
        INPUT: meeting room number;
OUTPUT: “Your meeting room is: meetingRoomNumber”;
SWITCH (meetingRoomNumber)
        CASE 1: maxCapacity = roomOne;
        CASE 2: maxCapacity = roomTwo;
        CASE 3: maxCapacity = roomThree;
        CASE 4: maxCapacity = roomFour;
        CASE 5: maxCapacity = roomFive;

OUTPUT: “Max Capacity of meetingRoomNumber is maxCapacity”;
OUTPUT: “How many people will be attending the meeting?”;
INPUT: numAttendees;

remainingCapacity = maxCapacity – numAttendees;
IF remainingCapacity >= 0
        OUTPUT: “Meeting may be held in meetingRoomNumber also you may invite
        remainingCapacity more people”;
ELSE
        OUTPUT: Meeting may not be held in meetingRoomNumber please select another room
        that will accommodate the remaining remainingCapacity people;
Return 0;
END
```

fireLaw Flow chart:

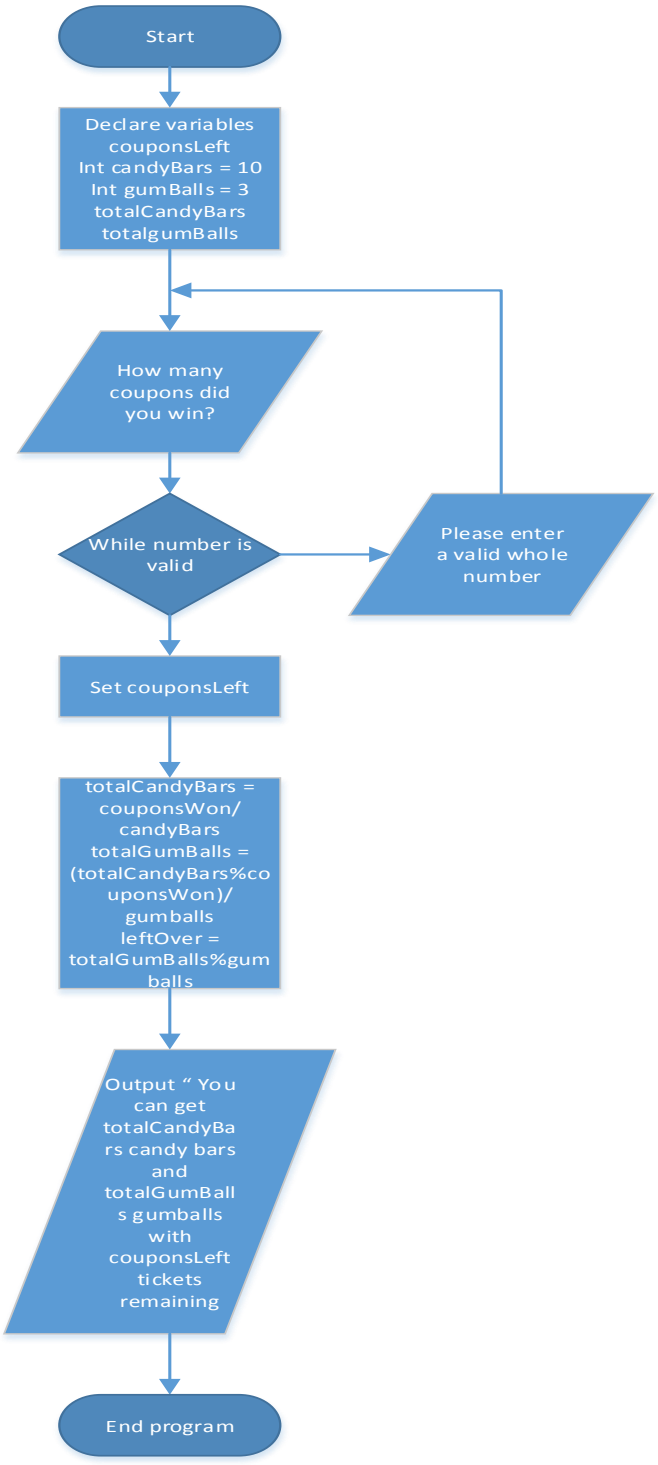


Program 2: arcade.cpp
Pseudocode

START int main()
DECLARE
    int couponsLeft = 0; totalCandyBars, totalGumBalls, couponsRemaining;
    const int candyBars = 10, gumBalls = 3;
OUTPUT: “How many coupons did you win?”;
WHILE: INPUT: couponsLeft is not valid char
    clear input;
    ignore input ;
    OUTPUT: “Please enter valid number;”
WHILE: Input invalid number
    OUTPUT: Please enter valid number;
    INPUT: couponsLeft;
totalCandyBars = couponsLeft/candyBars;
totalGumballs = (couponsLeft%candyBars)/gumBalls;
couponsRemaining = couponsLeft%gumBalls;

OUTPUT: “With couponsLeft you can get totalCandyBars candy bars and totalGumballs gum balls and have couponsRemaining coupons left;
RETURN 0;
END

Flow Chart arcade.cpp

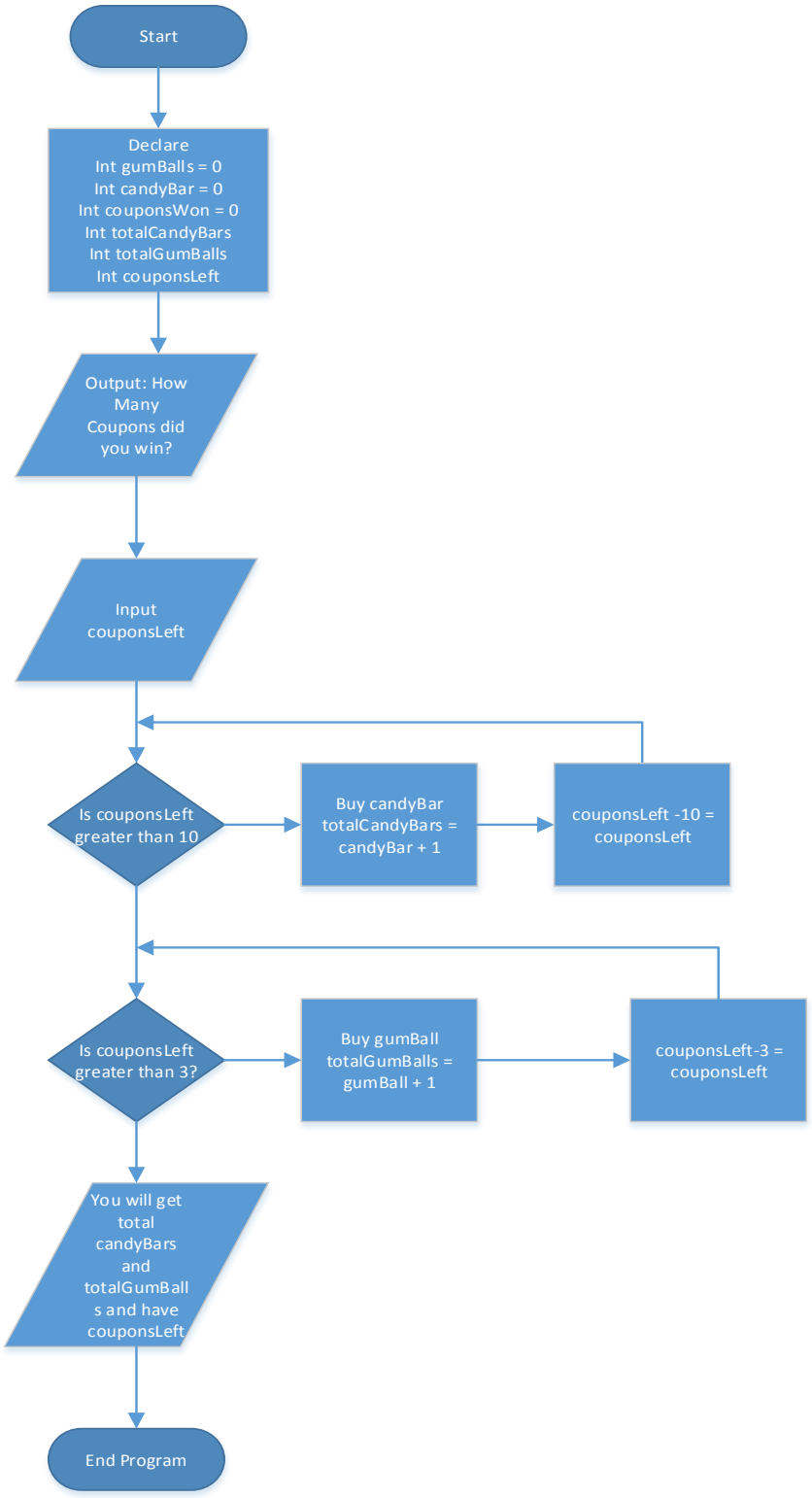


Program 3: arcade2.cpp

Pseudocode

START int main()
DECLARE
    int gumBalls = 0, candyBar = 0, couponsLeft;
OUTPUT : " How many coupons did you win?";
INPUT: couponsLeft;
WHILE: couponsLeft >= 10
    candyBar = candyBar +1;
    couponsLeft = couponsLeft -10;
WHILE: couponsLeft < 10 and couponsLeft > 2
    gumBalls = gumBalls + 1;
    couponsLeft = couponsLeft – 3;
OUTPUT: You have candyBar candy bars and gumBalls gum balls and couponsLeft coupons left
RETURN 0;
END

Flow Chart arcade2.cpp

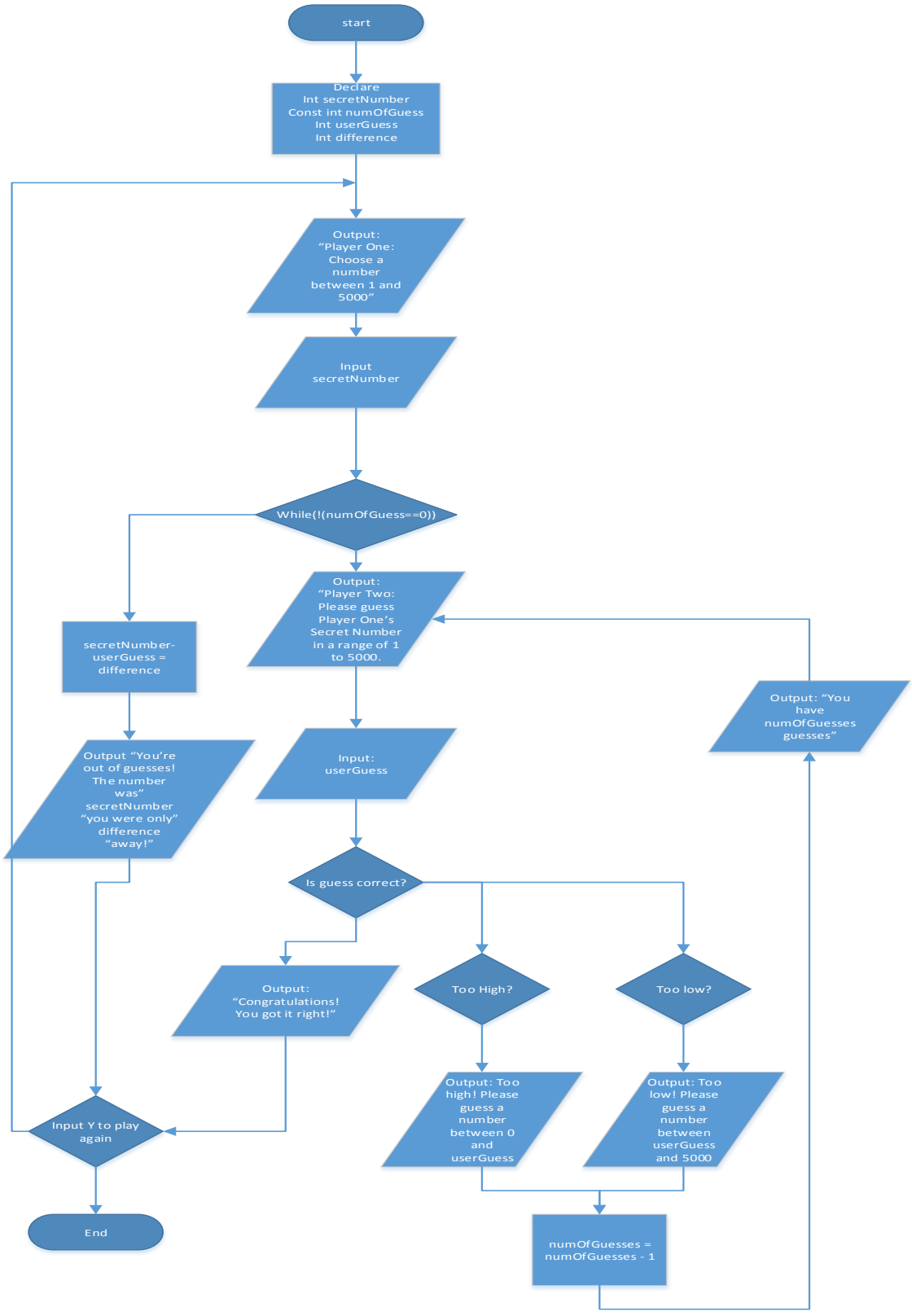

Start
Declare
Int gumBalls = 0
Int candyBar = 0
Int couponsWon = 0
Int totalCandyBars
Int totalGumBalls
Int couponsLeft
Output: How Many Coupons did you win?
Input couponsLeft
Is couponsLeft greater than 10
Buy candyBar totalCandyBars = candyBar + 1
couponsLeft -10 = couponsLeft
Is couponsLeft greater than 3?
Buy gumBall totalGumBalls = gumBall + 1
couponsLeft-3 = couponsLeft
You will get total candyBars and totalGumBalls and have couponsLeft
End Program

Programing Project: numGuess.cpp

Pseudocode

```
START int main()
DECLARE
        int secretNumber = 0, userGuess, difference, numOfGuess, lowNumber, highNumber;
        char again;
DO
        OUTPUT: “Player One: Please choose a number from 1 to 5000”;
        WHILE (INPUT: secretNumber = invalid number)
                Clear cin;
                Ignore cin;
                OUTPUT: “Please enter a valid number from 1 to 5000”;
        WHILE secretNumber > 5000 or secretNumber < 1
                OUTPUT: “Please enter a number from 1 to 5000”;
                INPUT: secretNumber;
        numOfGuess = 5;
        lowNumber = 1;
        highNumber = 5000;

        OUTPUT: “Player Two: Please guess the number from lowNumber and highNumber:”;

        WHILE numOfGuess does not equal 0
                INPUT: userGuess;
                WHILE userGuess isn’t a valid int
                        Clear cin
                        Ignore input
                OUTPUT: “Please enter a valid number from lowNumber to highNumber:”;
        numOfGuess = numOfGuess – 1;
        IF userGuess == secretNumber
                OUTPUT: “Congratulations you got it right!”;
                Break;
        ELSE IF userGuess>secretNumber
                OUTPUT: “userGuess is too freaking high”;
                OUTPUT: “You have numOfGuesses remaining”;
                OUTPUT: “Please enter a number between lowNumber and userGuess”;
        ELSE
                OUTPUT: “userGuess is too freaking low”;
                OUTPUT: “You have numOfGuesses remaining”;
                IF numOfGuess != 0
                        OUTPUT: “Please enter a number between userGuess and highNumber”;
                ELSE
                        break;
IF numOfGuess == 0
        Difference = secretNumber – userGuess;
        OUTPUT: “You’re out of guesses. The secret number is secretNumber. You were difference away!”;
WHILE again == Y or again == y
RETURN 0;
END
```

Programming Project numGuess.cpp Flow Chart


start
Declare
Int secretNumber
Const int numOfGuess
Int userGuess
Int difference
Output:
"Player One:
Choose a
number
between 1 and
5000"
Input
secretNumber
While(!(numOfGuess==0))
Output:
"Player Two:
Please guess
Player One's
Secret Number
in a range of 1
to 5000.
secretNumber-
userGuess =
difference
Output "You're
out of guesses!
The number
was"
secretNumber
"you were only"
difference
"away!"
Input:
userGuess
Output: "You
have
numOfGuesses
guesses"
Is guess correct?
Output:
"Congratulations!
You got it right!"
Too High?
Too low?
Output: Too
high! Please
guess a
number
between 0
and
userGuess
Output: Too
low! Please
guess a
number
between
userGuess
and 5000
Input Y to play
again
End
numOfGuesses =
numOfGuesses - 1

Testing

fireLaw.cpp

| Room # Entered | Expected Response | Actual Response |
|---|---|---|
| 1 | Capacity is 8 people | Expected |
| 2 | Capacity is 10 people | Expected |
| 3 | Capacity is 15 people | Expected |
| 4 | Capacity is 32 people | Expected |
| 5 | Capacity is 4 people | Expected |
| 0 | Please enter a valid room | Expected |
| -1 | Please enter a valid room | Expected |
| 300 | Please enter a valid room | Expected |
| D | Please enter a valid room number | Endless Loop |

| # of people attending room 2 (10 = max capacity) | Can you use the meeting hall | How many more people can attend | Expected result |
|---|---|---|---|
| 6 | YES | 4 | YES |
| 12 | NO | -2 | YES |
| -1 | YES | 11 | YES |
| D | YES | 10 | YES |

arcade.cpp

| # of Coupons | # of candy bars expected | Actual # of Candy Bars | # of gum balls expected | Actual # of gum balls | Expected # of coupons left over | Actual # of coupons left over |
|---|---|---|---|---|---|---|
| 10 | 1 | 1 | 0 | 0 | 0 | 1 Error |
| 20 | 2 | 2 | 0 | 0 | 0 | 2 Error |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| 33 | 3 | 3 | 1 | 1 | 0 | 0 |
| 37 | 3 | 3 | 2 | 2 | 1 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| -1 | Please enter valid # | | | | | |
| D | Please enter a valid # | | | | | |
| | | | | | | |

arcade2.cpp

| # of Coupons | # of candy bars expected | Actual # of Candy Bars | # of gum balls expected | Actual # of gum balls | Expected # of coupons left over | # of coupons left over |
|---|---|---|---|---|---|---|
| 10 | 1 | 1 | 0 | 0 | 0 | 0 |
| 20 | 2 | 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| 33 | 3 | 3 | 1 | 1 | 0 | 0 |
| 37 | 3 | 3 | 2 | 2 | 1 | 1 |
| -1 | 0 | 0 | 0 | 0 | -1 | -1 |
| D | 0 | 0 | 0 | 0 | 0 | 0 |

numGuess.cpp

| 1st # entered | Response expected | Actual response |
|---|---|---|
| 1 | Player 2 enter # | Expected |
| 5000 | Player 2 enter # | Expected |
| 5001 | Enter valid number in range | Expected |
| 0 | Enter valid number in range | Expected |
| F | Enter valid number | Expected |
| -300 | Enter valid | Expected |

| | | |
|---|---|---|
| | number in range | |

| Player two (guess 1) secret # = 5 | Expected response | New range | Actual response | Actual range |
|---|---|---|---|---|
| 5 | Congrats you're right | | | |
| 10 | Too high! | 1-10 | expected | Expected |
| 2 | Too Low | 2-5000 | expected | Expected |
| 5001 | Too High | 1-5001 | expected | Expected |
| D | Please enter a valid number | | Loop * see notes | |
| -1 | Too Low | -1-5000 | expected | expected |

| Player two (guess 5) secret # = 5 | Expected response | Actual response |
|---|---|---|
| 5 | Congrats you're right | Expected |
| 10 | Too high! Out of guesses you're 5 away | Expected |
| 1 | Too Low out of guesses you're 4 away | Expected |
| 5001 | Too High | Expected |
| D | Please enter a valid number | Loop * see notes |
| -1 | Too Low | Expected |

Reflection

The exercises initially were tough especially wrapping my head around the two's compliment discussion but after taking another look at it and going over it with the other students in my class I've got a good grasp of it. The lecture addressing addition of binary was very helpful as well in knowing what method would work best for me in converting decimal to binary. I did have an issues with the adding of the two binary numbers but converting to decimal and back certainly helped.

I made an effort to go above and beyond for the fireLaw program because I wanted to start using loops earlier and work on practicing the switch/case statement. Once I got an idea of what I'd like to do I had no problem with the logic to do so.

For the first arcade project I had to get the math down on paper initially to know how I'd like to do it. I did have some problems after I completed the project with some numbers giving the wrong number of remaining tickets after the program had run. For example, if I put down 20 tickets then I'd get two candy bars and 2 tickets remaining. I changed the equation to evaluate based on the difference in total tickets and tickets used instead of trying to use the gum ball equation to determine the remainder. I also initially tried to use two variables for the tickets won but determined it to unnecessary.

The difficulty with the second arcade equation was which loop to use. I thought I could possibly use a For loop but determined it would be easier to use the while as I did not know the exact number of iterations of the loop. While the logic of the program was a bit more demanding, the code itself was far simpler than the other arcade program. I'm guessing that was the purpose of doing it both ways.

Finally, this weeks project borrowed a bit from all of the assignments for the week. I had some difficulty with the looping of the entire program but found the do while loop used on page 261 of the text worked really well. I did include some error checking even though it was not required which worked great for the first user. For some reason the error checking did not work for the second user though and the program would go into a loop still or not have the output I wanted. I could have taken it out, but I would like to play with it further later to figure out what went wrong so I left the validation in the comments on the code. Beyond that issue I really enjoyed the challenge of the project and am happy with the result. I learned quite a bit from this program and its development.

Exercises

1. (2) Convert (feel free to just use paper, though I use a text editor for this sometimes) the following numbers from decimal to binary:

a. 3

3 = 2+1 = 2^1 + 2^0 = 000011

b. 7

7 = 4 + 2 + 1 = 2^2 + 2^1 + 2^0 = 000111

c. 10

10 = 8 + 2 = 2^3 + 0 + 2^1 + 0 = 001010

d. 50

50 = 32 + 16 + 2 = 2^5 + 2^4 + 0 + 0 + 2^1 + 0 = 110010

e. 94

94 = 64 + 16 + 8 + 4 + 2 = 2^6 + 0 + 2^4 + 2^3 + 2^2 + 2^1 + 0 = 01011110

f. 192

192 = 128 + 64 = 2^7 + 2^6 + 0 + 0 + 0 + 0 + 0 + 0 = 11000000

2. (2) Convert the following numbers from binary to decimal:

a. 10

10 = 2 ^ 1 + 0 = 2

b. 1110

1110 = 2^3 + 2^2 + 2^1 + 0 = 14

c. 111010

111010 = 2^5 + 2^4 + 2^3 + 0 + 2^1 + 0 = 32 + 16 + 8 + 0 + 2 = 58

d. 11100011

11100011 = 2^0 + 2^1 + 0 + 0 + 0 + 2^5 + 2^6 + 2^7 = 1 + 2 + 0 + 0 + 0 + 32 + 64 + 128 = 227

3. (2) Convert each of the decimal numbers from problem 1 above into two's compliment representation numbers with the minimum number of bits possible.

a. 3

3 = 2+1 = 2^1 + 2^0 = 000011
011

b. 7

7 = 4 + 2 + 1 = 2^2 + 2^1 + 2^0 = 000111
0111

c. 10

10 = 8 + 2 = 2^3 + 0 + 2^1 + 0 = 001010
01010

d. 50

50 = 32 + 16 + 2 = 2^5 + 2^4 + 0 + 0 + 2^1 + 0 = 110010
0110010

e. 94

94 = 64 + 16 + 8 + 4 + 2 = 2^6 + 0 + 2^4 + 2^3 + 2^2 + 2^1 + 0 = 01011110
01011110

f. 192

192 = 128 + 64 = 2^7 + 2^6 + 0 + 0 + 0 + 0 + 0 + 0 = 11000000
011000000

4. (2) 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).

a. -3

3 = 2+1 = 2^1 + 2^0 = 000011
011
100 + 1 = 101

b. -7

7 = 4 + 2 + 1 = 2^2 + 2^1 + 2^0 = 000111
0111
1000 + 1 = 1001

c. -10

10 = 8 + 2 = 2^3 + 0 + 2^1 + 0 = 001010
01010
10101 + 1 = 10110

d. -50

50 = 32 + 16 + 2 = 2^5 + 2^4 + 0 + 0 + 2^1 + 0 = 110010
0110010
1001101 + 1 = 1001110

e. -94

94 = 64 + 16 + 8 + 4 + 2 = 2^6 + 0 + 2^4 + 2^3 + 2^2 + 2^1 + 0 = 01011110

01011110
10100001 + 1 = 10100010

f. -192

192 = 128 + 64 = 2^7 + 2^6 + 0 + 0 + 0 + 0 + 0 + 0 = 11000000
011000000
100111111 + 1 = 101000000

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

1111
01100110 = 0 + 2^1 + 2^2 + 0 + 0 + 2^5 + 2^6 + 0 = 2 + 4 + 32 + 64 = 102
+01011100 = 0 + 0 + 2^2 + 2^3 + 2^4 + 0 + 2^6 + 0 = 4 + 8 + 16 + 64 = 92
11000010 194 = 128 + 64 + 2 = 2^7 + 2^6 + 0 + 0 + 0 + 0 + 2^1 + 0 = 11000010

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

10100001 = -1(2^0 + 0 + 0 + 0 + 0 + 2^5 + 0) = -1(1 + 64) = -65
+10101010 = -1( 0 + 2^1 + 0 + 2^3 + 0 + 2^5 + 0) = -1(2 + 8 + 64) = -74

-65 + -74 = -(139) = -(128 + 8 + 2 + 1) = -(2^7 + 0 + 0 + 0 + 2^3 + 0 + 2^1 + 2^0)
= -(010001011) = 101110100 + 1 = 101110101