COSC 221: Computer Organization I
Fall 2013

50/100

Programming Project #1: Signed Integer Representation

- *Documentation (15 points)*

README file		5	(5 points)
Judicious use of comments, white space, and identifiers		2	(5 points)
Modularity		2	(5 points)

- *Program Correctness (85 points)*

Correctly retrieves binary string input		3	(3 points)
Calculates and displays the decimal equivalent of the binary string as:			
signed magnitude	✓ ✓ ✓ ✓	8	(8 points)
one's complement	✓ ✗ ✓ ✗	4	(8 points)
two's complement	✓ ✗ ✓ ✗	4	(8 points)
excess-512 notation	✓ ✓ ✓ ✓	8	(8 points)
Correctly retrieves decimal integer input		2	(3 points)
Calculates and displays the equivalent 10-bit string representation using :			
signed magnitude	✗ ✗ ✗ ✗	1	(8 points)
one's complement	✗ ✗ ✓ ✗	2	(8 points)
two's complement	✗ ✗ ✓ ✗	2	(8 points)
excess-512 notation	✗ ✗ ✓ ✗	2	(8 points)
Implements input data validation		0	(5 points)
✓ Displays properly formatted and labeled output		5	(10 points)

Comments:

* error 'Number format Exception.'

* No option to continue "

```
//Zane Wonsey

package cosc.emich.edu;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner userInput = new Scanner(System.in);
boolean isRunning = true;

System.out.println("Please enter a 10 bit binary string or a decimal value

while (isRunning) {

if (userInput != null) {

String line = userInput.nextLine();

if (line.equals("x")) {
userInput.close();
System.exit(0);
}
int isLineInt = Integer.parseInt(line);

if (line.length() == 10) {
//Output lines
System.out.println("Decimal representation using one's complement: " + ones
System.out.println("Decimal representation using two's complement: " + twos
System.out.println("Decimal representation using signed magnitude: " + sig
System.out.println("Decimal representation using excess-512 notation: " +

} else if (isLineInt >= -512 && isLineInt <= 511) {

String x = decimalToBinary(isLineInt);

//Output lines
System.out.println("Binary representation using one's complement: " + ones(
System.out.println("Binary representation using two's complement: " + twos(
System.out.println("Decimal representation using signed magnitude: " + x);
System.out.println("Decimal representation using excess-512 notation: " +

} else {
System.err.println("Input entered incorrectly, please try again:");
}

} else {
isRunning = false;
}

}

}


/* Below are the methods used to evaluate the input
* from the user.
*
* They can take either a string or an integer as input
* while using and integer index to check what to do with
* the input it receives.
*/
public static String excess(int input, int index) {
if (index == 1) {
if (input < 0 ) {
return decimalToBinary(input + 512);
} else {
return decimalToBinary(input - 512);
}
} else {
if (input < 0 ) {
return Integer.toString(input + 512);
} else {
return Integer.toString(input - 512);
}
}
}
public static String signedMag(String data, int index) {
if (data.charAt(0) == '1') {
data = "0" + data.substring(1, 10);
return Integer.toString(binaryToDecimal(data) * -1);
} else {
data = "0" + data.substring(1, 10);
return Integer.toString(binaryToDecimal(data));
}
}

public static String onesComp(String binNumber, int index) {

String toReturn = "";
```

```

if (binNumber.charAt(0) =='0' && index == 1) {
return binNumber;
} else if (binNumber.charAt(0) =='0' && index == 0) {
return Integer.toString(binaryToDecimal(binNumber));
} else {
toReturn = toReturn + "1";
}
for (int i = 1; i <= 9; i++) {

if (binNumber.charAt(i) == '1') {

toReturn = toReturn + "0";

} else {

toReturn = toReturn + "1";

}

}

if (index == 1) {
return toReturn;
} else {
if (toReturn.charAt(0) == '0') {
return Integer.toString(binaryToDecimal(toReturn) * -1);
} else {
return Integer.toString(binaryToDecimal(toReturn));
}
}

}

public static String twosComp(String binNumber, int index) {
String toReturn = "";
int carry = 1;

if (binNumber.charAt(0) =='0' && index == 1) {
return binNumber;
}

for (int i = 9; i >= 0; i--) {
if (toReturn.length() != 10) {
if (binNumber.charAt(i) == '1' && carry == 1) {
toReturn = "0" + toReturn;
} else if (binNumber.charAt(i) == '1' && carry == 0) {
toReturn = "1" + toReturn;
} else if (binNumber.charAt(i) == '0' && carry == 1) {
toReturn = "1" + toReturn;
carry = 0;
} else if (binNumber.charAt(i) == '0' && carry == 0) {
toReturn = "0" + toReturn;
}
}
}

if (index == 1) {
return toReturn;
} else {
return Integer.toString(binaryToDecimal(toReturn));
}

}

public static int binaryToDecimal(String binNumber) {
double numberToReturn = 0;
int exponent = 0;
for (int n = 9; n >= 0; n--) {

if (binNumber.charAt(n) == '1') {

numberToReturn = numberToReturn + Math.pow(2, exponent);

}

exponent++;

}

return (int) numberToReturn;

}

public static String decimalToBinary(int data) {

int isNeg = data;

String binaryOutput = "";

while (data >= 1 && data <= 511 || data <= -1 && data >= -512) {

```

```
if (data % 2 == 0) {
binaryOutput = binaryOutput + "0";
} else {
binaryOutput = binaryOutput + "1";
}

data = data / 2;

}

while (binaryOutput.length() != 10) {

if (isNeg < 0 && binaryOutput.length() == 9) {
binaryOutput = "1" + binaryOutput;
} else {

binaryOutput = "0" + binaryOutput;

}
//System.out.println(binaryOutput);

}

return binaryOutput;

}

}
```