

**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

```
1 //Zane Wonsey
2
3 package cosc.emich.edu;
4
5 import java.util.Scanner;
6
7 public class Main {
8
9     public static void main(String[] args) {
10
11         Scanner userInput = new Scanner(System.in);
12         boolean isRunning = true;
13
14         System.out.println("Please enter a 10 bit binary string or a decimal value
15
16         while (isRunning) {
17
18             if (userInput != null) {
19
20                 String line = userInput.nextLine();
21
22                 if (line.equals("x")) {
23                     userInput.close();
24                     System.exit(0);
25                 }
26                 int isLineInt = Integer.parseInt(line);
27
28                 if (line.length() == 10) {
29                     //Output lines
30                     System.out.println("Decimal representation using one's complement: " + one's
31                     System.out.println("Decimal representation using two's complement: " + two's
32                     System.out.println("Decimal representation using signed magnitude: " + sign
33                     System.out.println("Decimal representation using excess-512 notation: " + (
34
35                 } else if (isLineInt >= -512 && isLineInt <= 511) {
36
37                     String x = decimalToBinary(isLineInt);
38
39                     //Output lines
40                     System.out.println("Binary representation using one's complement: " + one's
41                     System.out.println("Binary representation using two's complement: " + two's
42                     System.out.println("Decimal representation using signed magnitude: " + x);
43                     System.out.println("Decimal representation using excess-512 notation: " + (
44
45                 } else {
46                     System.err.println("Input entered incorrectly, please try again:");
```

```
47 }
48
49 } else {
50   isRunning = false;
51 }
52
53 }
54
55 }
56
57
58 /* Below are the methods used to evaluate the input
59 * from the user.
60 *
61 * They can take either a string or an integer as input
62 * while using an integer index to check what to do with
63 * the input it receives.
64 */
65 public static String excess(int input, int index) {
66   if (index == 1) {
67     if (input < 0 ) {
68       return decimalToBinary(input + 512);
69     } else {
70       return decimalToBinary(input - 512);
71     }
72   } else {
73     if (input < 0 ) {
74       return Integer.toString(input + 512);
75     } else {
76       return Integer.toString(input - 512);
77     }
78   }
79 }
80 public static String signedMag(String data, int index) {
81   if (data.charAt(0) == '1') {
82     data = "0" + data.substring(1, 10);
83     return Integer.toString(binaryToDecimal(data) * -1);
84   } else {
85     data = "0" + data.substring(1, 10);
86     return Integer.toString(binaryToDecimal(data));
87   }
88 }
89
90 public static String onesComp(String binNumber, int index) {
91
92   String toReturn = "";
```

```
93
94 if (binNumber.charAt(0) == '0' && index == 1) {
95     return binNumber;
96 } else if (binNumber.charAt(0) == '0' && index == 0) {
97     return Integer.toString(binaryToDecimal(binNumber));
98 } else {
99     toReturn = toReturn + "1";
100 }
101 for (int i = 1; i <= 9; i++) {
102
103     if (binNumber.charAt(i) == '1') {
104
105         toReturn = toReturn + "0";
106
107     } else {
108
109         toReturn = toReturn + "1";
110
111     }
112
113 }
114
115 if (index == 1) {
116     return toReturn;
117 } else {
118     if (toReturn.charAt(0) == '0') {
119         return Integer.toString(binaryToDecimal(toReturn) * -1);
120     } else {
121         return Integer.toString(binaryToDecimal(toReturn));
122     }
123 }
124
125 }
126
127 public static String twosComp(String binNumber, int index) {
128     String toReturn = "";
129     int carry = 1;
130
131     if (binNumber.charAt(0) == '0' && index == 1) {
132         return binNumber;
133     }
134
135     for (int i = 9; i >= 0; i--) {
136         if (toReturn.length() != 10) {
137             if (binNumber.charAt(i) == '1' && carry == 1) {
138                 toReturn = "0" + toReturn;
```



```
139 } else if (binNumber.charAt(i) == '1' && carry == 0) {
140 toReturn = "1" + toReturn;
141 } else if (binNumber.charAt(i) == '0' && carry == 1) {
142 toReturn = "1" + toReturn;
143 carry = 0;
144 } else if (binNumber.charAt(i) == '0' && carry == 0) {
145 toReturn = "0" + toReturn;
146 }
147 }
148 }
149
150 if (index == 1) {
151 return toReturn;
152 } else {
153 return Integer.toString(binaryToDecimal(toReturn));
154 }
155
156 }
157
158 public static int binaryToDecimal(String binNumber) {
159 double numberToReturn = 0;
160 int exponent = 0;
161 for (int n = 9; n >= 0; n--) {
162
163 if (binNumber.charAt(n) == '1') {
164
165 numberToReturn = numberToReturn + Math.pow(2, exponent);
166
167 }
168
169 exponent++;
170
171 }
172
173 return (int) numberToReturn;
174
175 }
176
177 public static String decimalToBinary(int data) {
178
179 int isNeg = data;
180
181 String binaryOutput = "";
182
183 while (data >= 1 && data <= 511 || data <= -1 && data >= -512) {
184
```

```
185 if (data % 2 == 0) {
186     binaryOutput = binaryOutput + "0";
187 } else {
188     binaryOutput = binaryOutput + "1";
189 }
190
191 data = data / 2;
192
193 }
194
195 while (binaryOutput.length() != 10) {
196
197     if (isNeg < 0 && binaryOutput.length() == 9) {
198         binaryOutput = "1" + binaryOutput;
199     } else {
200
201         binaryOutput = "0" + binaryOutput;
202     }
203 }
204 //System.out.println(binaryOutput);
205
206 }
207
208 return binaryOutput;
209
210 }
211
212 }
213
214
```