Nathan Brooks

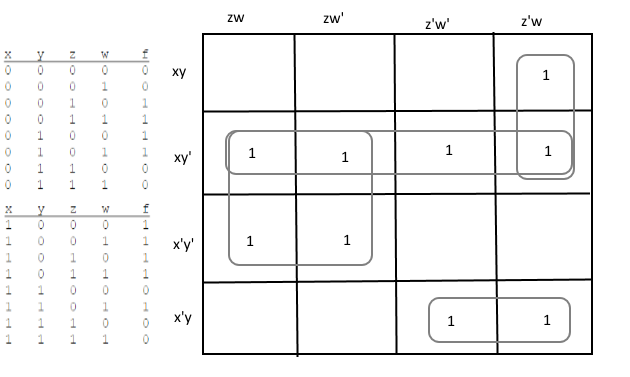
P2Part2

Part A

1 // Program Proj2A.java  
 2 // Function: f is the truth value of a logical proposition.  
 3   
 4 import java.io.\*;  
 5   
 6 public class P2PartA  
 7 {  
 8 public static void main(String args[])  
 9 {  
10 System.out.println(LProp('F', 'F', 'F'));  
11 System.out.println(LProp('F', 'F', 'T'));  
12 System.out.println(LProp('F', 'T', 'F'));  
13 System.out.println(LProp('F', 'T', 'T'));  
14 System.out.println(LProp('T', 'F', 'F'));  
15 System.out.println(LProp('T', 'F', 'T'));  
16 System.out.println(LProp('T', 'T', 'F'));  
17 System.out.println(LProp('T', 'T', 'T'));  
18 }  
19   
20 private static char LProp(char p, char q, char r)  
21 {  
22 // Logical expression  
23 //f(p,q,r) = (~q AND r AND ~p) OR (~(r OR ~p))  
24   
25 return ORlogic(ANDlogic(ANDlogic(NOTlogic(q), r), NOTlogic(p)), NOTlogic(ORlogic(r, NOTlogic(p))));  
26 }  
27   
28 private static char ANDlogic(char p, char q)  
29 {  
30 // Logical AND function  
31 char f = 'F';  
32 if(p == 'T' && q == 'T')  
33 {  
34 f = 'T';  
35 }  
36 return f;  
37 }  
38   
39 private static char ORlogic(char p, char q)  
40 {  
41 // Logical OR function   
42 char f = 'F';  
43 if(p == 'T' || q == 'T')  
44 {  
45 f = 'T';  
46 }  
47 return f;   
48 }  
49   
50 private static char NOTlogic(char p)  
51 {  
52 // Logical NOT function  
53 if(p == 'T')  
54 {  
55 return 'F';  
56 }  
57 else  
58 {  
59 return 'T';  
60 }  
61 }  
62 } // end class



Part B



f(x, y, z, w)

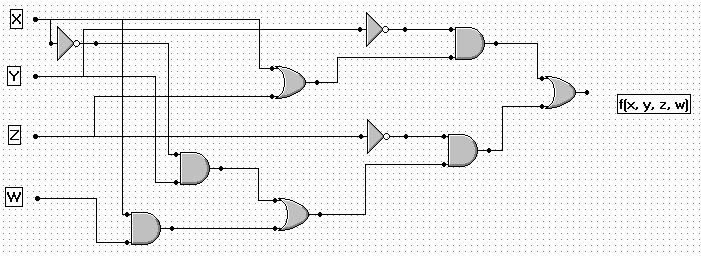
= y’z + xy’ + z’wx + x’yz’

= y’(z + x) + z’(wx + x’y)

NOT 3

AND 4

OR 3



1 import java.io.\*;  
 2   
 3 public class P2PartB4  
 4 {  
 5 public static void main(String args[])  
 6 {  
 7 System.out.println(LCircuit(0, 0, 0, 0));  
 8 System.out.println(LCircuit(0, 0, 0, 1));  
 9 System.out.println(LCircuit(0, 0, 1, 0));  
10 System.out.println(LCircuit(0, 0, 1, 1));  
11 System.out.println(LCircuit(0, 1, 0, 0));  
12 System.out.println(LCircuit(0, 1, 0, 1));  
13 System.out.println(LCircuit(0, 1, 1, 0));  
14 System.out.println(LCircuit(0, 1, 1, 1));  
15 System.out.println(LCircuit(1, 0, 0, 0));  
16 System.out.println(LCircuit(1, 0, 0, 1));  
17 System.out.println(LCircuit(1, 0, 1, 0));  
18 System.out.println(LCircuit(1, 0, 1, 1));  
19 System.out.println(LCircuit(1, 1, 0, 0));  
20 System.out.println(LCircuit(1, 1, 0, 1));  
21 System.out.println(LCircuit(1, 1, 1, 0));  
22 System.out.println(LCircuit(1, 1, 1, 1));  
23 }  
24   
 private static int LCircuit(int x, int y, int z, int w)  
26 {  
27 // Logical expression  
28   
29 //f(x, y, z, w) = y’(z + x) + z’(wx + x’y)  
30   
31 return ORgate( ANDgate(NOTgate(y), ORgate(z, x)), ANDgate(NOTgate(z), ORgate(ANDgate(w, x), ANDgate(NOTgate(x), y))));  
32

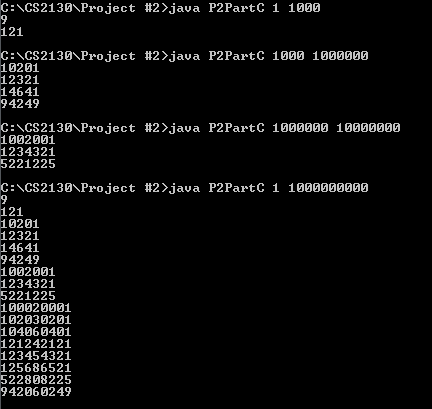
33   
34 }  
35   
36 private static int ANDgate(int p, int q)  
37 {  
38 // Logical AND function  
39 int f = 0;  
40 if(p == 1 && q == 1)  
41 {  
42 f = 1;  
43 }  
44 return f;  
45 }  
46   
47 private static int ORgate(int p, int q)  
48 {  
49 // Logical OR function   
50 char f = 0;  
51 if(p == 1 || q == 1)  
52 {  
53 f = 1;  
54 }  
55 return f;   
56 }  
57   
58 private static int NOTgate(int p)  
59 {  
60 // Logical NOT function  
61 if(p == 1)  
62 {  
63 return 0;  
64 }  
65 else  
66 {  
67 return 1;  
68 }  
69 }  
70 } // end class



Part C:

1 import java.io.\*;  
 2   
 3 public class P2PartC  
 4 {  
 5 public static void main(String args[])  
 6 {  
 7 long N1, N2;  
 8 if(args.length >= 2){  
 9 N1 = Long.parseLong(args[0]);  
 10 N2 = Long.parseLong(args[1]);   
 11 }  
 12 else  
 13 {  
 14 N1 = 0;  
 15 N2 = 1;  
 16 }  
 17   
 18 // check all the numbers between the given values  
 19 for (long i=0; i < N2 - N1; i++)  
 20 {  
 21 if(KNumber(i+N1) == true)  
 22 {  
 23 System.out.println(i + N1);  
 24 }  
 25 }  
 26   
 27 return;  
 28 }  
 29   
 30 //------------------------------------  
 31   
 32 private static boolean OddInt(long X)  
 33 {  
 34 if(X % 2 == 1)  
 35 {  
 36 return true;  
 37 }  
 38 else  
 39 {  
 40 return false;  
 41 }  
 42 }  
 43   
 44 //------------------------------------  
 45   
 46 private static boolean SquareInt(long X)  
 47 {  
 48 for(int i=1; i<(X/2); i += 2) // only check odds  
 49 {  
 50 if(i \* i == X)  
 51 {  
 52 return true;  
 53 }  
 54 else if(i \* i > X) // no point continuing to run this past X  
 55 {  
 56 return false;  
 57 }  
 58 }  
 59 return false;  
 60 }  
 61   
 62 //------------------------------------  
 63   
 64 private static boolean SymmetricInt(long X)  
 65 {  
 66 if(X == reverseDigits(X))  
 67 {  
 68 return true;  
 69 }  
 70 else  
 71 {  
 72 return false;  
 73 }   
 74 }  
 75   
 76 //------------------------------------  
 77   
 78 private static boolean KNumber(long X)  
 79 {  
 80 if(OddInt(X))  
 81 {  
 82 if(SymmetricInt(X))  
 83 {  
 84 if(SquareInt(X))  
 85 {  
 86 return true;  
 87 }  
 88 else  
 89 {  
 90 return false;  
 91 }  
 92 }  
 93 else  
 94 {  
 95 return false;  
 96 }   
 97 }  
 98 else  
 99 {  
100 return false;  
101 }  
102 }  
103   
104 //-----------------------------------  
105 // I wrote this function myself in the last assignment for this class

// I figured I might as well use it, it worked so well  
106 //-----------------------------------  
107   
108 public static long reverseDigits(long x)  
109 {  
110 long y = 0;  
111 int intLength = 0; // to test how many digits x has  
112   
113 for(int i=1; i<=x; i\*=10)  
114 {  
115 if(x >= i)  
116 {  
117 intLength++; // count how many digits x has  
118 }  
119 }  
120   
121 for(int i=0; i<intLength; i++)  
122 {  
123 // shift result to the left one  
124 y \*= 10;  
125   
126 // add the remainder from dividing by 10  
127 y += x % 10;  
128   
129 // shift x right one and truncates off the decimal because it is a long  
130 x /= 10;  
131 }  
132   
133 return y;  
134 }  
135   
136 } // end class



Yes there are more numbers, I ran it again up to 1 billion

I’m a little curious to see what would happen up to 1 trillion

But I would have to use something bigger than a long data type

Part D:

1 import java.io.\*;  
 2   
 3 public class P2PartD // TLogic  
 4 {  
 5 public static void main(String args[])  
 6 {  
 7 System.out.println(ternaryFunction('F', 'F', 'F'));  
 8 System.out.println(ternaryFunction('F', 'F', 'T'));  
 9 System.out.println(ternaryFunction('F', 'F', 'U'));  
 10   
 11 System.out.println(ternaryFunction('F', 'T', 'F'));  
 12 System.out.println(ternaryFunction('F', 'T', 'T'));  
 13 System.out.println(ternaryFunction('F', 'T', 'U'));  
 14   
 15 System.out.println(ternaryFunction('F', 'U', 'F'));  
 16 System.out.println(ternaryFunction('F', 'U', 'T'));  
 17 System.out.println(ternaryFunction('F', 'U', 'U'));  
 18   
 19 System.out.println(ternaryFunction('T', 'F', 'F'));  
 20 System.out.println(ternaryFunction('T', 'F', 'T'));  
 21 System.out.println(ternaryFunction('T', 'F', 'U'));  
 22   
 23 System.out.println(ternaryFunction('T', 'T', 'F'));  
 24 System.out.println(ternaryFunction('T', 'T', 'T'));  
 25 System.out.println(ternaryFunction('T', 'T', 'U'));  
 26   
 27 System.out.println(ternaryFunction('T', 'U', 'F'));  
 28 System.out.println(ternaryFunction('T', 'U', 'T'));  
 29 System.out.println(ternaryFunction('T', 'U', 'U'));   
 30   
 31 System.out.println(ternaryFunction('U', 'F', 'F'));  
 32 System.out.println(ternaryFunction('U', 'F', 'T'));  
 33 System.out.println(ternaryFunction('U', 'F', 'U'));  
 34   
 35 System.out.println(ternaryFunction('U', 'T', 'F'));  
 36 System.out.println(ternaryFunction('U', 'T', 'T'));  
 37 System.out.println(ternaryFunction('U', 'T', 'U'));  
 38   
 39 System.out.println(ternaryFunction('U', 'U', 'F'));  
 40 System.out.println(ternaryFunction('U', 'U', 'T'));  
 41 System.out.println(ternaryFunction('U', 'U', 'U'));   
 42 }  
 43   
44 private static char ternaryFunction(char p, char q, char r)  
 45 {  
 46 // Logical expression  
 47 //f(p,q,r) = (~q AND r AND ~p) OR (~(r OR ~p))  
 48   
 49 return TOR(TAND(TAND(TNOT(q), r), TNOT(p)), TNOT(TOR(r, TNOT(p))));  
 50 }  
 51   
 52 private static char TAND(char p, char q)  
 53 {  
 54 // Logical AND function  
 55 // false as first parameter  
 56 if(p == 'F' && q == 'F')  
 57 {  
 58 return 'F';  
 59 }  
 60 else if(p == 'F' && q == 'T')  
 61 {  
 62 return 'F';  
 63 }  
 64 else if(p == 'F' && p == 'U')  
 65 {  
 66 return 'F';  
 67 }  
 68   
 69 // true as first parameter  
 70 else if(p == 'T' && p == 'F')  
 71 {  
 72 return 'F';  
 73 }  
 74 else if(p == 'T' && p == 'T')  
 75 {  
 76 return 'T';  
 77 }  
 78 else if(p == 'T' && p == 'U')  
 79 {  
 80 return 'U';  
 81 }  
 82   
 83 // unknown as first parameter  
 84 else if(p == 'U' && p == 'F')  
 85 {  
 86 return 'F';  
 87 }  
 88 else if(p == 'U' && p == 'T')  
 89 {  
 90 return 'U';  
 91 }  
 92 else  
 93 {  
 94 return 'U';  
 95 }  
 96 }  
 97   
 98 private static char TOR(char p, char q)  
 99 {  
100 // Logical OR function   
101 // false as first parameter  
102 if(p == 'F' || p == 'F')  
103 {  
104 return 'F';  
105 }  
106 else if(p == 'F' || p == 'T')  
107 {  
108 return 'T';  
109 }  
110 else if(p == 'F' || p == 'U')  
111 {  
112 return 'U';  
113 }  
114   
115 // true as first parameter  
116 else if(p == 'T' || p == 'F')  
117 {  
118 return 'T';  
119 }  
120 else if(p == 'T' || p == 'T')  
121 {  
122 return 'T';  
123 }  
124 else if(p == 'T' || p == 'U')  
125 {  
126 return 'T';  
127 }  
128   
129 // unknown as first parameter  
130 else if(p == 'U' || q == 'F')  
131 {  
132 return 'U';  
133 }  
134 else if(p == 'U' || p == 'T')  
135 {  
136 return 'T';  
137 }  
138 else  
139 {  
140 return 'U';  
141 }   
142 }  
143   
144 private static char TNOT(char p)  
145 {  
146 // Logical NOT function  
147 if(p == 'U')  
148 {  
149 return 'U';  
150 }  
151 else if(p == 'T')  
152 {  
153 return 'F';  
154 }  
155 else  
156 {  
157 return 'T';  
158 }  
159 }  
160 } // end class

