```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Globalization;
 4  using System.IO;
 5  using System.Linq;
 6
 7  namespace Zadania._2015;
 8
 9  public class D07Z01 : IZadanie
10  {
11      private Int32 ZapaloneZarowki;
12      private List<string> Komendy;
13      private List<Operacja> Operacje;
14      private SortedDictionary<string, UInt16> Przewody;
15
16      public D07Z01(bool daneTestowe = false)
17      {
18          this.Komendy = new();
19          this.Przewody = new();
20          this.Operacje = new();
21
22          this.ZapaloneZarowki = 0;
23          FileStream fs = new(daneTestowe ? ".\\Dane\\2015\\07\
              \proba.txt" : ".\\Dane\\2015\\07\\dane.txt", FileMode.Open,
              FileAccess.Read);
24          StreamReader sr = new(fs);
25          string linia;
26
27          while((linia = sr.ReadLine()) is not null)
28          {
29              this.Komendy.Add(linia);
30          }
31
32          sr.Close(); fs.Close();
33      }
34
35      public void RozwiazanieZadania()
36      {
37          this.WczytajPrzewody();
38
39          this.Wypelnij();
40      }
41
42      private void WczytajPrzewody()
43      {
44          string[] przewod;
45          string[] dzialanie;
46
47          foreach(string komenda in this.Komendy)
48          {
49              przewod = komenda.Split("->").Select(p => p.Trim()).ToArray
                ();
50
```

```
51              if (!this.Przewody.ContainsKey(przewod[1]) &&
                  UInt16.TryParse(przewod[0], out UInt16 wartosc))
52              {
53                  this.Przewody.Add(przewod[1], wartosc);
54                  continue;
55              }
56
57              if ((dzialanie = przewod[0].Split(' ')).Length == 1)
58              {
59                  this.Operacje.Add(new(dzialanie[0], przewod[1],
                      string.Empty, string.Empty));
60                  continue;
61              }
62
63              if ((dzialanie = przewod[0].Split(' ')).Length == 2)
64              {
65                  this.Operacje.Add(new(dzialanie[0], dzialanie[1],
                      przewod[1], string.Empty));
66                  continue;
67              }
68
69              if ((dzialanie = przewod[0].Split(' ')).Length == 3)
70              {
71                  this.Operacje.Add(new(dzialanie[0], dzialanie[1],
                      dzialanie[2], przewod[1]));
72              }
73          }
74      }
75
76      public string PokazRozwiazanie()
77      {
78          return this.ZapaloneZarowki.ToString("N0",
                CultureInfo.CreateSpecificCulture("pl-PL"));
79      }
80
81      private void Wypelnij()
82      {
83          bool wartoscB = false, wynikB = false, wartoscLB = false,
                wartoscPB = false;
84          UInt16 wartosc = 0, wynik = 0, wartoscL = 0,  wartoscP = 0;
85          Operacja operacja;
86
87          while(this.Operacje.Count > 0)
88          {
89              for(int wierszI = 0; wierszI < this.Operacje.Count; wierszI
                    ++)
90              {
91                  wartoscLB = wartoscPB = wartoscB = wynikB = false;
92
93                  operacja = this.Operacje[wierszI];
94
95                  if(operacja.c.Equals(string.Empty))
96                  {
```

```
97                      if(UInt16.TryParse(operacja.a, out wynik))
98                      {
99                          this.Przewody.Add(operacja.b, wynik);
100                     }
101
102                     continue;
103                 }
104
105                 if (operacja.a.Equals("NOT"))
106                 {
107                     if (this.Przewody.ContainsKey(operacja.b))
108                     {
109                         this.Przewody.TryGetValue(operacja.b, out
                              wartosc);
110                         wartoscB = true;
111                         wynikB = false;
112                     }
113
114                     if (this.Przewody.ContainsKey(operacja.c))
115                     {
116                         this.Przewody.TryGetValue(operacja.c, out
                              wynik);
117                         wartoscB = false;
118                         wynikB = true;
119                     }
120
121                     switch (operacja.a, wartoscB, wynikB)
122                     {
123                         case ("NOT", true, false):
124                             this.Przewody.Add(operacja.c,
                                  Convert.ToUInt16(UInt16.MaxValue -
                                  wartosc));
125                             this.Operacje.RemoveAt(wierszI);
126                             wierszI--;
127                             continue;
128                         case ("NOT", false, true):
129                             this.Przewody.Add(operacja.b,
                                  Convert.ToUInt16(UInt16.MaxValue -
                                  wynik));
130                             this.Operacje.RemoveAt(wierszI);
131                             wierszI--;
132                             continue;
133                     }
134
135                     continue;
136                 }
137
138                 if (!operacja.a.Equals("NOT"))
139                 {
140                     if (this.Przewody.ContainsKey(operacja.a))
141                     {
142                         this.Przewody.TryGetValue(operacja.a, out
                              wartoscL);
```

```
143                         wartoscLB = true;
144                     }
145
146                     if (this.Przewody.ContainsKey(operacja.c))
147                     {
148                         this.Przewody.TryGetValue(operacja.c, out
                              wartoscP);
149                         wartoscPB = true;
150                     }
151
152                     if (this.Przewody.ContainsKey(operacja.wynik))
153                     {
154                         this.Przewody.TryGetValue(operacja.wynik, out
                              wynik);
155                         wynikB = true;
156                     }
157
158                     switch (operacja.b, wartoscLB, wartoscPB, wynikB)
159                     {
160                         case ("LSHIFT", true, false, false):
161                             wynik = Convert.ToUInt16(wartoscL <<
                                  Convert.ToUInt16(operacja.c));
162                             this.Przewody.Add(operacja.wynik, wynik);
163                             this.Operacje.RemoveAt(wierszI);
164                             wierszI--;
165                             continue;
166                         case ("RSHIFT", true, false, false):
167                             wynik = Convert.ToUInt16(wartoscL >>
                                  Convert.ToUInt16(operacja.c));
168                             this.Przewody.Add(operacja.wynik, wynik);
169                             this.Operacje.RemoveAt(wierszI);
170                             wierszI--;
171                             continue;
172                         case ("AND", true, true, false):
173                             wynik = Convert.ToUInt16(wartoscL &
                                  wartoscP);
174                             this.Przewody.Add(operacja.wynik, wynik);
175                             this.Operacje.RemoveAt(wierszI);
176                             wierszI--;
177                             continue;
178                         case ("OR", true, true, false):
179                             wynik = Convert.ToUInt16(wartoscL |
                                  wartoscP);
180                             this.Przewody.Add(operacja.wynik, wynik);
181                             this.Operacje.RemoveAt(wierszI);
182                             wierszI--;
183                             continue;
184                     }
185                 }
186             }
187         }
188     }
189
```

```
190     private record Operacja(string a, string b, string c, string
          wynik);
191 }
```