```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace INFINIT
8  {
9      class infinit
10     {
11         List<int> number = new List<int>();
12         public char Sign;
13         public infinit(string num)
14         {
15             //სტრინგს დავშლით და ავაწყობთ ინტების ლისტს
16             int i = 0;
17             //გავარკვევთ გადმოცემული რიცხვის ნიშანს
18             if (num[0] == '-')
19             {
20                 Sign = '-';
21                 i++;
22             }
23             else Sign ='+';
24             //თუ წინ მინუსი აქვს ინტებად დაშლას დაიწყებს მეორე, ხოლო წინააღმდეგ შემთხვევაში პირველივე ელემენტიდან
25             for (; i < num.Length; i++)
26             {
27                 number.Add(int.Parse(num[i].ToString()));
28             }
29         }
30         infinit(List<int> a, char s)
31         {
32             //ზოგიერთი ოპერაციისთვის კლასის შიგნით დაგვჭირდება ასეთი კონსტრუქტორიც, რომელსაც ლისტს და ნიშანს გადავცემთ
33             number = a;
34             Sign = s;
35         }
36         public static infinit operator *(infinit inf1, infinit inf2)
37         {
38             char s;
39             //იმის მიხედვით თუ რა ნიშნები აქვთ გადაცემულ რიცხვებს, განვსაზღვრავთ შედეგის ნიშანს
40             if ((inf1.Sign == '+' && inf2.Sign == '+') || (inf1.Sign == '-' && inf2.Sign == '-')) s = '+';
41             else s = '-';
42             infinit ret = Mult(inf1.number, inf2.number);
43             ret.Sign = s;
44             //თუ მინუსიანი მივიღეთ მაგრამ რიცხვი ნულია, ნიშანში ვუწერთი პლიუსს რათა შედეგის გამოტანისას "-0" არ დაგვიწეროს
45             if (ret.ToString()[0] == '-' && ret.ToString()[1] == '0') ret.Sign = '+';
46             return ret;
47         }
48         public static infinit operator -(infinit inf1, infinit inf2_2)
49         {
50             //აქ მთავარი არის ნიშნის განსაზღვრა, ხოლო გამოკლებას მიმატებაზე
```

```
                მანიპულირებითაც შევძლებთ
            infinit inf2 = new infinit(inf2_2.number, inf2_2.Sign);
            if (inf2.Sign == '+') inf2.Sign = '-';
            else inf2.Sign = '+';
            infinit tmp = inf1 + inf2;
            return new infinit(tmp.number, tmp.Sign);
        }
        public static infinit operator +(infinit inf1, infinit inf2)
        {
            List<int> temp = new List<int>();
            if (inf1.Sign == '-') //თუ პირველი რიცხვი უარყოფითია
            {
                if (inf2.Sign == '+') // ხოლო მეორე დადებითი
                {
                    if (compare1(inf1.number, inf2.number) == 1) //და ნიშნის გარეშე პირველი მეტია
                    {
                        //პირველს გამოაკლდება მეორე და დაეწერება მინუს ნიშანი
                        temp = Sub(inf1.number, inf2.number);
                        return new infinit(temp, '-');
                    }
                    else //ხოლო, თუ ნიშნის გარეშე მეორეა მეტი, ან ტოლია
                    {
                        //მეორეს გამოაკლდება პირველი და დაეწერება + ნიშანი
                        temp = Sub(inf2.number, inf1.number);
                        return new infinit(temp, '+');
                    }
                }
                else //თუ მეორე უარყოფითია
                {
                    //შეიკრიბება და დაეწერება მინუს ნიშანი
                    temp = Sum(inf1.number, inf2.number);
                    return new infinit(temp, '-');
                }
            }
            else //თუ პირველი რიცხვი დადებითია
            {
                if (inf2.Sign == '+') //და მეორეც დადებითი
                {
                    //შეიკრიბება და დაეწერება + ნიშანი
                    temp = Sum(inf1.number, inf2.number);
                    return new infinit(temp, '+');
                }
                else //მეორე კი უარყოფითი
                {
                    if(compare1(inf1.number, inf2.number) == -1) //და ნიშნის გარეშე მეორე მეტია პირველზე
                    {
                        //მეორეს გამოაკლდება პირველი და ნიშანი იქნება -
                        temp = Sub(inf2.number, inf1.number);
                        return new infinit(temp, '-');
                    }
                    else //პირველი მეტია ან ტოლია მეორეზე
                    {
                        //პირველს გამოაკლდება მეორე და ნიშანი იქნება +
```

```
103                         temp = Sub(inf1.number, inf2.number);
104                         return new infinit(temp, '+');
105                     }
106                 }
107             }
108         }
109         static infinit Mult(List<int> x, List<int> y)
110         {
111             List<int> a;
112             List<int> b;
113             //გავარკვიოთ რომელია და a-ში ჩავწეროთ უფრო გრძელი რიცხვი
114             if (compare1(x, y) == 1)
115             {
116                 a = new List<int>(x);
117                 b = new List<int>(y);
118             }
119             else
120             {
121                 a = new List<int>(y);
122                 b = new List<int>(x);
123             }
124             //გამრავლების დროს ქვეშმიწერის ხაზს ქვემოთ ვიღებთ რამდენიმე
                რიცხვს (მატრიცის მსგავსად) რომელიც შემდეგ უნდა შევკრიბოთ
125             //ამ რიცხვების შესანახად გამოვიყენოთ ლისტების ლისტი
126             List<List<int>> result = new List<List<int>>();
127             for (int i = b.Count - 1; i >= 0; i--) //დავიწყოთ გამრავლება
                ბოლოდან
128             {
129                 //b რიცხვის ყოველი ციფრი გავამრავლოთ a-ს ყოველ ციფრზე
130                 List<int> temp = new List<int>(); //თითოეული მომავალში
                    შესაკრების შესანახად
131                 //წანაცვლებამდე ჩავსვათ ნულები
132                 for (int j = 0; j < b.Count - 1 - i; j++)
133                 {
134                     temp.Add(0);
135                 }
136                 int c = 0; //დამახსოვრებულისთვის (რომელიც თავიდან ნულია)
137                 for (int j = a.Count - 1; j >= 0; j--)
138                 {
139                     temp.Insert(0, (b[i] * a[j] + c) % 10); //მნიშვნელობა
                        რომელიც იწერება
140                     c = (b[i] * a[j] + c) / 10; //დამახსოვრებული
                        მნიშვნელობა
141                 }
142                 if (c != 0) temp.Insert(0, c); //თუ ბოლოს კიდევ დაგვრჩება
                    დამახსოვრებული მნიშვნელობა, ჩავამატოთ წინ
143                 result.Add(temp);
144             }
145             infinit ret = new infinit("0"); //შევქმნათ რიცხვი რომელიც
                თავიდან ნულის ტოლია
146             //და მისი მეშვეობით შევკრიბოთ გამრავლების შედეგად მიღებული
                შესაკრები რიცხვები (მატრიცა)
147             for(int i = 0; i < result.Count; i++)
148             {
149                 infinit temp = new infinit(result[i], '+');
150                 ret = ret + temp;
```

```
151             }
152             return ret;
153         }
154         public static List<int> Sum(List<int> x, List<int> y)
155         {
156             List<int> a = new List<int>(x);
157             List<int> b = new List<int>(y);
158 
159             List<int> ret = new List<int>();
160             //ვპოულობთ არის თუ არა სიგრძეებს შორის განსხვავება, თუ ასეა მის მნიშვნელობას (რამდენით განსხვავდება)
161             //ვინახავთ dif ში, და ამის მიხედვით იმას, რომლის სიგრძეც ნაკლებია, მეორეს სიგრძემდე შესავსებად წინ ჩავუმატებთ 0-ებს
162             if (a.Count > b.Count)
163             {
164                 int dif = a.Count - b.Count;
165                 for (int i = 0; i < dif; i++) b.Insert(0, 0);
166             }
167             else if (b.Count > a.Count)
168             {
169                 int dif = b.Count - a.Count;
170                 for (int i = 0; i < dif; i++) a.Insert(0, 0);
171             }
172             int c = 0; //დამახსოვრებული მნიშვნელობის შესანახად (რომელიც თავიდან ნულია)
173             for(int i = a.Count - 1; i >= 0; i--) //შეკრებას ვიწყებთ ბოლოდან
174             {
175                 ret.Insert(0, (a[i] + b[i] + c) % 10); //ჩავამატოთ შედეგის ლისტში ის მნიშვნელობა რასაც ქვეშმიწერით შეკრების დროს ვწერთ
176                 if (a[i] + b[i] + c > 9) c = 1; //c-ცვლადში კი შევინახოთ დამახსოვრებული მნიშვნელობა თუ საჭიროა (1-ზე მეტი ვერ იქნება)
177                 else c = 0;  //წინააღმდეგ შემთხვევაში c = 0
178             }
179             if (c == 1) ret.Insert(0, 1); //ყველაზე წინა თანრიგების მიმატებისას თუ ისევ დაგვრჩა დამახსოვრებული მნიშვნელობა, ჩავამატოთ
180             return ret;
181         }
182         static List<int> Sub(List<int> x, List<int> y)
183         {
184             //როდესაც აქ გადავცემ ლისტებს, ნიშნის გარეშე პირველი მეტი უნდა იყოს მეორეზე
185             List<int> a = new List<int>(x);
186             List<int> b = new List<int>(y);
187 
188             List<int> ret = new List<int>();
189             //აქ კი ვამოწმებთ, პირველი სიგრძითაც მეტი ხომ არაა მეორეზე და თუ ასეა მეორეს წინ ნულებით შევავსებთ პირველის სიგრძემდე
190             if (a.Count > b.Count)
191             {
192                 int dif = a.Count - b.Count;
193                 for (int i = 0; i < dif; i++) b.Insert(0, 0);
194             }
195             for (int i = a.Count - 1; i >= 0; i--) //გამოკლებას ვიწყებთ
```

```
                ბოლოდან
196             {
197                 if (a[i] < b[i]) //თუ ზედას ქვედა "არ აკლდება"
198                 {
199                     int j = i - 1;
200                     while (true) //მაშინ გადავდივართ მარცხნივ და ვეძებთ ციფრს რომლისგანაც ვისესხებთ
201                     {
202                         if (a[j] == 0) a[j] = 9; //როდესაც სესხებას გავაკეთებთ თუ, გამსესხებელ რიცხვსა და მსესხებელს შორის არის 0-ები, მაშინ იქ უნდა ჩავწეროთ 9
203                         else
204                         {
205                             a[j] = a[j] - 1; //და თუ არანულოვან ციფრს მივადგებით, მისგან ვისესხებთ, რაც იმას ნიშნავს რომ, ის შემცირდება 1-ით
206                             break;
207                         }
208                         j--;
209                     }
210                     ret.Insert(0, 10 + a[i] - b[i]); //სესხების გამო, a[i]-ის ნაცვლად a[i]+10 -ს დააკლდება b[i]
211                 }
212                 else
213                 {
214                     ret.Insert(0, a[i] - b[i]); //თუ სესხება არ გვჭირდება, პირდაპირ დააკლდება
215                 }
216             }
217             while ((ret[0] == 0) && ret.Count != 1) ret.Remove(0); //თუ წინ ზედმეტი 0-ები დაგვრჩება, ამოვიღებთ
218             return ret;
219         }
220         static int compare1(List<int> a, List<int> b)
221         {
222             if (a.Count > b.Count) return 1;
223             else if (b.Count > a.Count) return -1;
224             else
225             {
226                 //თუ სიგრძეები ტოლია, მაშინ დაიწყება ერთნაირი თანრიგის ციფრების შედარება
227                 for(int i = 0; i < a.Count; i++)
228                 {
229                     if (a[i] > b[i]) return 1;
230                     else if (a[i] < b[i]) return -1;
231                 }
232             }
233             return 0;
234         }
235         public override string ToString()
236         {
237             //ჯერ განვსაზღვრავთ ნიშანს, თუ მინუსია, მაშინ დავწერთ ხოლო თუ პლიუსია - არა.
238             string s = "";
239             if (Sign == '-') s += "-";
240             //ჩვეულებრივ, ვბეჭდავთ ლისტს
```

```
241             for(int i = 0; i < number.Count; i++)
242             {
243                 s += number[i].ToString();
244             }
245             return s;
246         }
247     }
248 }
249
```