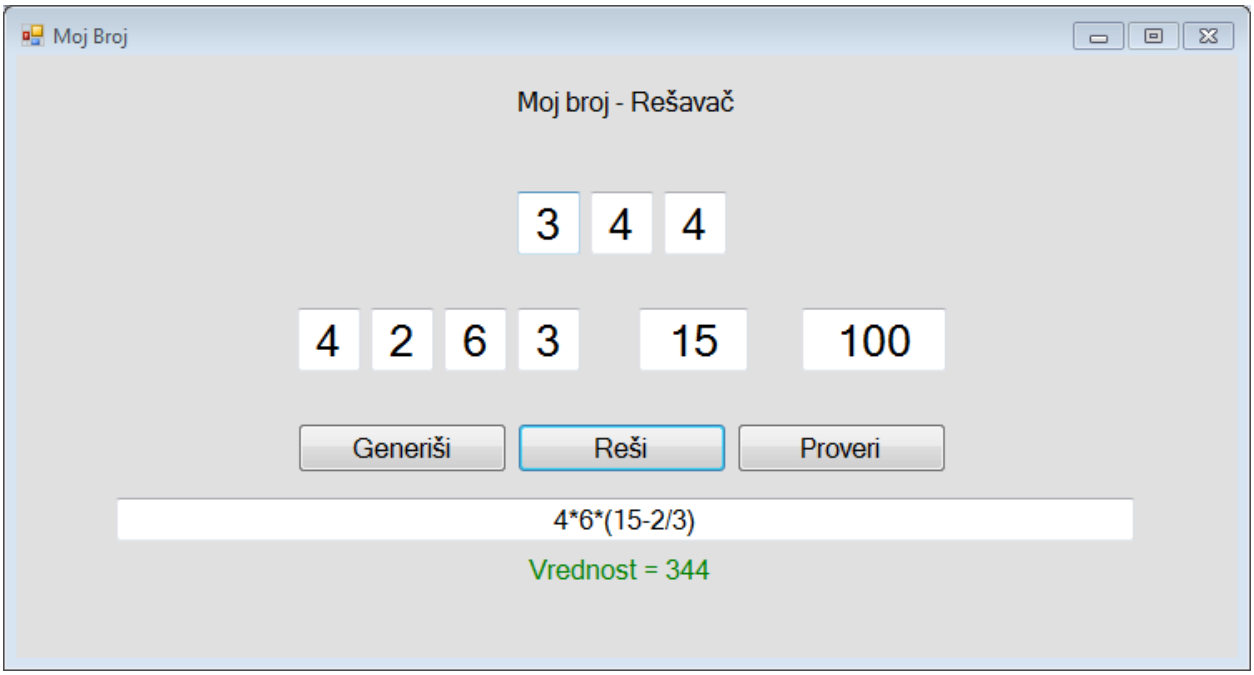
Rad u Microsoft Visual Studio 2013



Forma

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MojBroj
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            ponudjeni = new int[6];
            generisi();
            nacrtaj();
        }

        private int trazeni;
        private int[] ponudjeni;
        private bool prekidac = true;

        private void generisi() {
            Random rng = new Random();
            trazeni = (int)Math.Floor(rng.NextDouble() * 1000);
```

```
            for (int i = 0; i < 4; i++) {
                ponudjeni[i] = (int)Math.Floor(1 + rng.NextDouble() * 9);
            }
            ponudjeni[4] = (int)(10 + 5 * Math.Floor(3 * rng.NextDouble()));
            ponudjeni[5] = (int)(25 + 25 * Math.Floor(4 * rng.NextDouble()));
        }

        private void resetuj() {
            textBox10.Text = "";
            textBox10.BackColor = Color.White;
            label2.Text = "Vrednost = ?";
            label2.ForeColor = Color.Black;
        }

        private void promeni(object sender, EventArgs e) {
            if (prekidac) {
                if (!int.TryParse(textBox1.Text + textBox2.Text + textBox3.Text, out
trazeni)) {
                    trazeni = 0;
                }
                if (trazeni < 0 || trazeni > 999) trazeni = 0;

                if (!int.TryParse(textBox4.Text, out ponudjeni[0])) {
                    ponudjeni[0] = 1;
                }

                if (!int.TryParse(textBox5.Text, out ponudjeni[1])) {
                    ponudjeni[1] = 1;
                }

                if (!int.TryParse(textBox6.Text, out ponudjeni[2])) {
                    ponudjeni[2] = 1;
                }

                if (!int.TryParse(textBox7.Text, out ponudjeni[3])) {
                    ponudjeni[3] = 1;
                }

                if (!int.TryParse(textBox8.Text, out ponudjeni[4])) {
                    ponudjeni[4] = 10;
                }

                if (!int.TryParse(textBox9.Text, out ponudjeni[5])) {
                    ponudjeni[5] = 25;
                }

                for (int i = 0; i < 4; i++) {
                    if (ponudjeni[i] < 1 || ponudjeni[i] > 9) ponudjeni[i] = 1;
                }

                if (ponudjeni[4] < 1 || ponudjeni[4] > 99) ponudjeni[4] = 1;

                if (ponudjeni[5] < 1 || ponudjeni[5] > 999) ponudjeni[5] = 1;

                nacrtaj();
                resetuj();
            }
        }
```

```
        private void nacrtaj() {
            prekidac = false;
            //ciljni broj
            textBox1.Text = (trazeni / 100).ToString();
            textBox2.Text = (trazeni / 10 % 10).ToString();
            textBox3.Text = (trazeni % 10).ToString();

            //cifre
            textBox4.Text = ponudjeni[0].ToString();
            textBox5.Text = ponudjeni[1].ToString();
            textBox6.Text = ponudjeni[2].ToString();
            textBox7.Text = ponudjeni[3].ToString();

            //broj 10-15-20
            textBox8.Text = ponudjeni[4].ToString();

            //broj 25-50-75-100
            textBox9.Text = ponudjeni[5].ToString();

            prekidac = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            generisi();
            nacrtaj();
            resetuj();
        }

        private void button3_Click(object sender, EventArgs e) {
            Atom[] parsovan = Funkcije.ParsirajInfiks("(" + textBox10.Text + ")");
            Atom rv = Funkcije.racunajInfiks(parsovan);
            if (rv.getOp() == '!') {
                textBox10.BackColor = Color.Red;
                label2.ForeColor = Color.Black;
                label2.Text = "Neispravan izraz!";
            } else if (!Funkcije.podskup(parsovan, ponudjeni)){
                label2.Text = "Koristite nedozvoljene brojeve!";
                label2.ForeColor = Color.Red;
            } else {
                textBox10.BackColor = Color.White;
                if (rv.getVr().q != 1) {
                    label2.Text = "Vrednost nije ceo broj!";
                    label2.ForeColor = Color.Red;
                } else {
                    label2.Text = "Vrednost = " + rv.getVr().p;
                    label2.ForeColor = rv.getVr().p == trazeni ? Color.Green :
Color.Black;
                }
            }
        }

        private void textBox10_TextChanged(object sender, EventArgs e) {
            textBox10.BackColor = Color.White;
            label2.Text = "Vrednost = ?";
            label2.ForeColor = Color.Black;
        }
```

```
        private void textBox10_KeyDown(object sender, KeyEventArgs e) {
            if (e.KeyCode == Keys.Enter) {
                button3_Click(null, null);
            }
        }

        private void button2_Click(object sender, EventArgs e) {
            Izraz resenje = Funkcije.nadjiNajblizi(trazeni, ponudjeni);
            textBox10.Text = resenje.infiks;
            button3_Click(null, null);
        }
    }
}
```

Funkcije

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MojBroj{
    class Razlomak {
        public int p, q;
        private static int nzd(int a, int b){
            while (b!=0){
                int t = b;
                b = a%b;
                a = t;
            }
            return a;
        }

        private void skrati(){
            if (q<0){
                p *= -1;
                q *= -1;
            }
            int r;
            if (p<0) r = nzd(p,q);
            else r = nzd(-p,q);
            p /= r;
            q /= r;
        }

        public static implicit operator Razlomak(int p) {
            return new Razlomak(p, 1);
        }

        public Razlomak(int p=0, int q=1) {
            this.p = p;
            this.q = q;
```

```
    skrati();
}

public static Razlomak operator+(Razlomak a, Razlomak b){
    Razlomak x = new Razlomak();
    x.p = a.p*b.q + a.q*b.p;
    x.q = a.q*b.q;
    x.skrati();
    return x;
}

public static Razlomak operator-(Razlomak a, Razlomak b){
    Razlomak x = new Razlomak();
    x.p = a.p*b.q - a.q*b.p;
    x.q = a.q*b.q;
    x.skrati();
    return x;
}

public static Razlomak operator *(Razlomak a, Razlomak b) {
    Razlomak x = new Razlomak();
    x.p = a.p * b.p;
    x.q = a.q * b.q;
    x.skrati();
    return x;
}

public static Razlomak operator /(Razlomak a, Razlomak b) {
    Razlomak x = new Razlomak();
    x.p = a.p * b.q;
    x.q = a.q * b.p;
    x.skrati();
    return x;
}

public static bool operator <(Razlomak a, Razlomak b) {
    return a.p * b.q < b.p * a.q;
}

public static bool operator >(Razlomak a, Razlomak b) {
    return b < a;
}

/*
public static bool operator ==(Razlomak a, Razlomak b) {
    return a.p * b.q == b.p * a.q;
}

public static bool operator !=(Razlomak a, Razlomak b) {
    return a.p * b.q != b.p * a.q;
}
*/
public static bool operator <=(Razlomak a, Razlomak b) {
    return a.p * b.q <= b.p * a.q;
}

public static bool operator >=(Razlomak a, Razlomak b) {
    return a.p * b.q >= b.p * a.q;
```

```
    }

    public static Razlomak abs(Razlomak x) {
        if (x.p < 0) x.p = -x.p;
        return x;
    }

    public bool ceo(){
        return q==1;
    }
}

class Atom {
    private char operacija;
    private Razlomak vrednost;

    public Atom(Razlomak vrednost){
        this.vrednost = vrednost;
        operacija = '\0';
    }

    public Atom(char operacija){
        this.operacija = operacija;
        vrednost = 0;
    }

    public bool jeOp() {
        return operacija != '\0';
    }

    public char getOp() {
        return operacija;
    }

    public Razlomak getVr() {
        return vrednost;
    }
}

class Izraz : IComparable<Izraz>{
    public String infiks;
    public char spoljnaOperacija; //element {+,-,*,/,e}. e ako je u pitanju samo broj
    public Razlomak vrednost;
    public int brojeva;

    public Izraz(Razlomak vred){
        vrednost = vred;
        spoljnaOperacija = 'e';
        infiks = vred.p.ToString(); //pretpostavlja se da je q=1
        brojeva = 1;
    }

    public override String ToString() {
        return infiks;
    }

    public Izraz(Izraz a, char op, Izraz b) {
        //pretpostavlja se da je b != 0 ako je u pitanju deljenje
```

```
            if (op=='+'){
                spoljnaOperacija = '+';
                infiks = a.infiks + op + b.infiks; //zagrade nisu potrebne
                vrednost = a.vrednost + b.vrednost;
            } else
            if (op=='-'){
                spoljnaOperacija = '-';
                String b1 = b.infiks;
                //jedan interesantan granicni slucaj
                if (b.spoljnaOperacija == '-' || b.spoljnaOperacija == '+') {
                    b1 = "(" + b1 + ")";
                }
                infiks = a.infiks + op + b1;
                vrednost = a.vrednost - b.vrednost;
            } else {
                //zajednicko i za * i za /
                spoljnaOperacija = op;
                String a1, b1;

                if (a.spoljnaOperacija == '+' || a.spoljnaOperacija == '-') a1 = "(" +
a.infiks + ")";
                else a1 = a.infiks;

                if (b.spoljnaOperacija == '+' || b.spoljnaOperacija == '-') b1 = "(" +
b.infiks + ")";
                else b1 = b.infiks;

                //jos jedan interesantan granicni slucaj
                if (b.spoljnaOperacija == '/' && op == '/') {
                    b1 = "(" + b.infiks + ")";
                }

                infiks = a1 + op + b1;

                if (op=='*'){
                    vrednost = a.vrednost * b.vrednost;
                } else {
                    vrednost = a.vrednost / b.vrednost;
                }
            }
            brojeva = a.brojeva + b.brojeva;
        }

        public static bool operator< (Izraz a, Izraz b){
            return a.vrednost < b.vrednost;
        }

        public static bool operator >(Izraz a, Izraz b) {
            return a.vrednost > b.vrednost;
        }

        public int CompareTo(Izraz b) {
            if (this < b) return -1;
            if (this > b) return 1;
            return 0;
        }
    }
```

```
class MySet {
    private SortedSet<Izraz> skup;
    public MySet() {
        skup = new SortedSet<Izraz>();
    }
    public void Add(Izraz x) {
        if (skup.Contains(x)) {
            Izraz stari = skup.GetViewBetween(x, x).ToList()[0];
            skup.Remove(x);
            skup.Add(Funkcije.bolji(stari, x));
        } else {
            skup.Add(x);
        }
    }

    public List<Izraz> ToList() {
        return skup.ToList();
    }
}

class Funkcije{
    public static Atom[] ParsirajInfiks(String izraz){
        List<Atom> lista = new List<Atom>();
        int broj = -1;
        for (int i = 0; i < izraz.Length; i++){
            if ("()*/+-".IndexOf(izraz[i]) >= 0) {
                if (broj != -1){
                    lista.Add(new Atom(broj));
                    broj = -1;
                }
                lista.Add(new Atom(izraz[i]));
            } else if (Char.IsDigit(izraz[i])) {
                if (broj == -1) broj = izraz[i] - '0';
                else broj = 10 * broj + izraz[i] - '0';
            } else {
                ; // ignorišemo sve ostale karaktere
            }
        }
        if (broj != -1) lista.Add(new Atom(broj));
        return lista.ToArray();
    }

    public static Atom racunajInfiks(Atom[] niz) {

        Atom[] stek = new Atom[niz.Length];
        int visina = 0;

        for (int i = 0; i < niz.Length; i++) {
            stek[visina++] = niz[i];

            bool imaSmisla = true;

            while (imaSmisla) {
                imaSmisla = false;

                //Ovo parče koda izgleda rogobatno i neelegantno
                //međutim demonstrira da je moguće procesirati izraz u Infiks
                //notaciji u linearnom vremenu
```

```
                    if (visina >= 3 && stek[visina - 2].getOp() == '*'
                                    && !stek[visina - 1].jeOp()
                                    && !stek[visina - 3].jeOp()) {
                        stek[visina - 3] = new Atom(stek[visina - 3].getVr() *
stek[visina - 1].getVr());
                        visina -= 2;
                        imaSmisla = true;
                    } else if (visina >= 3 && stek[visina - 2].getOp() == '/'
                                    && !stek[visina - 1].jeOp()
                                    && !stek[visina - 3].jeOp()) {
                        stek[visina - 3] = new Atom(stek[visina - 3].getVr() /
stek[visina - 1].getVr());
                        visina -= 2;
                        imaSmisla = true;
                    } else if (visina >= 5 && stek[visina - 4].getOp() == '+'
                                           && stek[visina - 2].getOp() != '\0'
                                           && "+-".IndexOf(stek[visina - 2].getOp()) >= 0
                                           && !stek[visina - 1].jeOp()
                                           && !stek[visina - 3].jeOp()
                                           && !stek[visina - 5].jeOp()) {
                        stek[visina - 5] = new Atom(stek[visina - 5].getVr() +
stek[visina - 3].getVr());
                        stek[visina - 4] = stek[visina - 2];
                        stek[visina - 3] = stek[visina - 1];
                        visina -= 2;
                        imaSmisla = true;
                    } else if (visina >= 5 && stek[visina - 4].getOp() == '-'
                                             && stek[visina - 2].getOp() != '\0'
                                             && "+-".IndexOf(stek[visina - 2].getOp()) >=
0
                                             && !stek[visina - 1].jeOp()
                                             && !stek[visina - 3].jeOp()
                                             && !stek[visina - 5].jeOp()) {
                        stek[visina - 5] = new Atom(stek[visina - 5].getVr() -
stek[visina - 3].getVr());
                        stek[visina - 4] = stek[visina - 2];
                        stek[visina - 3] = stek[visina - 1];
                        visina -= 2;
                        imaSmisla = true;
                    } else if (visina >= 5 && stek[visina - 1].getOp() == ')'
                                           && !stek[visina - 2].jeOp()
                                           && stek[visina - 3].getOp() == '+'
                                           && !stek[visina - 4].jeOp()
                                           && stek[visina - 5].getOp() == '(') {
                        stek[visina - 5] = new Atom(stek[visina - 4].getVr() +
stek[visina - 2].getVr());
                        visina -= 4;
                        imaSmisla = true;
                    } else if (visina >= 5 && stek[visina - 1].getOp() == ')'
                                             && !stek[visina - 2].jeOp()
                                             && stek[visina - 3].getOp() == '-'
                                             && !stek[visina - 4].jeOp()
                                             && stek[visina - 5].getOp() == '(') {
                        stek[visina - 5] = new Atom(stek[visina - 4].getVr() -
stek[visina - 2].getVr());
                        visina -= 4;
                        imaSmisla = true;
```

```
                } else if (visina >= 3 && stek[visina - 1].getOp() == ')'
                                       && !stek[visina - 2].jeOp()
                                       && stek[visina - 3].getOp() == '(') {
                    stek[visina - 3] = stek[visina - 2];
                    visina -= 2;
                    imaSmisla = true;
                }
            }
        }

        if (visina != 1) {
            return new Atom('!');
        } else {
            return new Atom(stek[0].getVr());
        }
    }

    public static Izraz bolji(Izraz a, Izraz b) {
        if (a.brojeva < b.brojeva) {
            return a;
        } else if (b.brojeva < a.brojeva) {
            return b;
        } else {
            return a.infiks.Length < b.infiks.Length ? a : b;
        }
    }

    public static Izraz nadjiNajblizi(int trazeni, int[] ponudjeni) {
        int n = ponudjeni.Length;
        List<Izraz>[] a = new List<Izraz>[1 << n];
        for (int i = 0; i < (1 << n); i++) a[i] = new List<Izraz>();
            for (int i = 0; i < n; i++) {
                a[1 << i].Add(new Izraz(ponudjeni[i]));
            }
        for (int i = 3; i < (1 << n); i++) {
            if (i != (i & (-i))) { //bit hakovanje
                MySet tmp = new MySet();
                for (int j = 1; j < (1 << n); j++){
                    int k = i - j;
                    if ((i & j) == j && (i & k) == k) {
                        foreach (var lexp in a[j]) {
                            foreach (var rexp in a[k]) {
                                if (j < k) tmp.Add(new Izraz(lexp, '+', rexp));
                                if (lexp.vrednost >= rexp.vrednost) tmp.Add(new
Izraz(lexp, '-', rexp));
                                if (j < k) tmp.Add(new Izraz(lexp, '*', rexp));
                                if (rexp.vrednost.p != 0) {
                                    tmp.Add(new Izraz(lexp, '/', rexp));
                                }
                            }
                        }
                    }
                }
                a[i] = tmp.ToList();
            }
        }
        Izraz naj = a[1][0];
        Razlomak trazen = trazeni;
```

```
            for (int i = 1; i < (1 << n); i++) {
                foreach(var exp in a[i]) if (exp.vrednost.q==1){ //if ceo
                    Razlomak razlika = Razlomak.abs(exp.vrednost - trazen) -
                                       Razlomak.abs(naj.vrednost - trazen);
                    if (razlika.p < 0){
                        naj = exp;
                    } else if (razlika.p == 0) {
                        naj = bolji(naj, exp);
                    }
                }
            }
            return naj;
        }

        public static bool podskup(Atom[] pod, int[] skup) {
            List<int> skupKaoLista = skup.ToList();
            for (int i = 0; i < pod.Length; i++) {
                if (!pod[i].jeOp()) {
                    if (!skupKaoLista.Remove(pod[i].getVr().p)) {
                        return false;
                    }
                }
            }
            return true;
        }
    }
}
```