ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА №6

Студент Синцов М.Ю

Группа 6201-120303D

Руководитель Борисов Д. С.

САМАРА 2025

Задание 1

```
public static double integrate(Function f, double lefftX, double rightX, double
discret){
    double sum = 0;
    if (discret <= 0) {
        throw new IllegalArgumentException("Шаг интегрирования должен быть
положительным");
    }

    if (lefftX >= rightX) {
        throw new IllegalArgumentException("Левая граница должна быть меньше
правой");
    }
    if(f.getLeftDomainBorder()>lefftX||f.getRightDomainBorder()<rightX)
        throw new IllegalArgumentException("Интервал интегрирования выходит
за границы оюласти определения функции");

    if(lefftX+discret>rightX)
        discret = rightX-lefftX;

    for(;lefftX<rightX;lefftX+=discret){
        if(lefftX+discret>=rightX)
            discret = rightX-lefftX;

        sum +=
(f.getFunctionValue(lefftX)+f.getFunctionValue(lefftX+discret))*discret/2;
    }
    return sum;
}
```

Задание 2

```
package functions.threads;

import functions.Function;


public class Task {
    private Function f;
    private double left;
    private double right;
    private double step;
    private int tasks;
```

```
    public Task(int tasksCount) {
        this.tasks = tasksCount;
    }

    // Синхронизированные геттеры и сеттеры
    public synchronized void setF(Function function) {
        this.f = function;
    }

    public synchronized Function getF() {
        return f;
    }

    public synchronized void setLeft(double leftBorder) {
        this.left = leftBorder;
    }

    public synchronized double getLeft() {
        return left;
    }

    public synchronized void setRight(double rightBorder) {
        this.right = rightBorder;
    }

    public synchronized double getRight() {
        return right;
    }

    public synchronized void setStep(double integrationStep) {
        this.step = integrationStep;
    }

    public synchronized double getStep() {
        return step;
    }

    public synchronized int getTasks() {
        return tasks;
    }
}
```

```
private static void nonThread(){
    System.out.println("Метод, реализующий последовательную версию
```

```
программы");
    Task task = new Task(100);
    for(int i = 0; i<task.getTasks();i++) {
        double base = Math.random() * 100;
        double left = Math.random()*100;
        double right = (Math.random()*(200-100+1))+100;
        double step = Math.random();
        Function f = new Log(base);
        task.setF(f);
        task.setLeft(left);
        task.setRight(right);
        task.setStep(step);
        System.out.println("Source: "+ task.getLeft() + " " + task.getRight() + " " +
task.getStep());
        double integrated =
Functions.integrate(task.getF(),task.getLeft(),task.getRight(),task.getStep());
        System.out.println("Result: " + task.getLeft() + " " + task.getRight() + " " +
task.getStep() + " " + integrated);
        System.out.println();
    }
}
```

1. БАЗОВОЕ ИНТЕГРИРОВАНИЕ

1.1 Интеграл экспоненты на [0, 1]:

Теоретическое значение: 1,718281828459045

Результаты интегрирования с разными шагами:

Шаг	Интеграл	Погрешность	Относит. погр.
0,5000	1,753931092464826	0,035649264005780	2,07e-02
0,1000	1,719713491389315	0,001431662930270	8,33e-04
0,0100	1,718296147450417	0,000014318991372	8,33e-06
0,0010	1,718281971649194	0,000000143190149	8,33e-08
0,0001	1,718281829891118	0,000000001432073	8,33e-10

1.2 Интеграл линейной функции f(x) = 2x на [0, 3]:

Теоретическое значение: 9,000000

Численное значение: 9,0000000000

Погрешность: 0,0000000000

Относительная погрешность: 9,81e-14

2. ПОИСК ОПТИМАЛЬНОГО ШАГА

2.1 Поиск шага для точности 1e-7 (7 знаков после запятой):

Найденный шаг: 0,0003906250

Значение интеграла: 1,718281850308207

Фактическая погрешность: 0,000000021849162

Требуемая погрешность: 0,000000100000000

Условие выполнено: ДА

2.2 Проверка с разными начальными шагами:

Нач. шаг	Оптимальный шаг	Погрешность
1,000	0,0002441406	0,000000008534800
0,500	0,0002441406	0,000000008534800
0,100	0,0003906250	0,000000021849162
0,050	0,0003906250	0,000000021849162

2.3 Зависимость шага от требуемой точности:

Точность	Оптимальный шаг	Количество отрезков
1e-03	0,0250000000	40
1e-04	0,0125000000	80

1e-05	0,0031250000	320
1e-06	0,0007812500	1280
1e-07	0,0003906250	2560
1e-08	0,0000976563	10240

Задание 3

```
package functions.threads;

import functions.Function;
import functions.basic.Log;
import java.util.Random;

public class SimpleGenerator implements Runnable {
    private Task task;
    private Random random = new Random();

    public SimpleGenerator(Task task) {
        this.task = task;
    }

    @Override
    public void run() {
        try {
            for (int i = 0; i < task.getTasks(); i++) {
                double base = 1 + random.nextDouble() * 9;
                double leftBorder = random.nextDouble() * 100;
                double rightBorder = 100 + random.nextDouble() * 100;
                double step = random.nextDouble();

                synchronized (task) {
                    task.setF(new Log(base));
                    task.setLeft(leftBorder);
                    task.setRight(rightBorder);
                    task.setStep(step);
                }

                System.out.printf("Source %.4f %.4f %.4f%n", leftBorder, rightBorder,
step);
                Thread.sleep(10);
            }
        } catch (InterruptedException e) {
            System.out.println("Generator was interrupted");
        }
```

```
    }
}
```

```
package functions.threads;
import functions.Function;
import static functions.Functions.integrate;

public class SimpleIntegrator implements Runnable {
    private Task task;

    public SimpleIntegrator(Task task) {
        this.task = task;
    }

    @Override
    public void run() {
        int processed = 0;

        while (processed < task.getTasks()) {
            Function function = null;
            double left = 0;
            double right = 0;
            double step = 0;

            synchronized (task) {
                function = task.getF();
                left = task.getLeft();
                right = task.getRight();
                step = task.getStep();
                task.setF(null);
            }

            if (function != null) {
                processed++;
                try {
                    double result = integrate(function, left, right, step);
                    System.out.printf("Result %.4f %.4f %.4f %.8f%n", left, right, step,
result);
                } catch (Exception e) {
                    System.out.println("Integrator error: " + e.getMessage());
                }
            }
            else {
                try {
```

```
                Thread.sleep(5);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                break;
            }
        }
    }
    System.out.println("Integrator завершен");
  }
}
```

2. ПРОСТАЯ МНОГОПОТОЧНАЯ ВЕРСИЯ

Метод реализующий потоковую версию программы

Source 28,3036 118,0410 0,0891

Result 28,3036 118,0410 0,0891 213,45657395

Source 27,8577 144,9485 0,3849

Result 27,8577 144,9485 0,3849 269,13389314

Source 8,1061 191,3493 0,5448

Result 8,1061 191,3493 0,5448 364,10674313

Source 32,2413 161,4934 0,4142

Result 32,2413 161,4934 0,4142 253,89019676

Source 32,2663 125,3414 0,8733

Result 32,2663 125,3414 0,8733 382,63403741

Source 75,8961 131,0725 0,8560

Result 75,8961 131,0725 0,8560 118,25186452

Source 46,9541 179,6490 0,2041

Result 46,9541 179,6490 0,2041 719,31333307

Source 20,4666 170,6960 0,0276

Result 20,4666 170,6960 0,0276 2066,22298178

Source 48,9141 180,6098 0,9873

Result 48,9141 180,6098 0,9873 539,32733033

Source 14,4760 113,5973 0,1878

Result 14,4760 113,5973 0,1878 190,16790342

Source 43,6098 174,2583 0,1783

Result 43,6098 174,2583 0,1783 309,52134234

Source 24,9349 146,3567 0,8902

Result 24,9349 146,3567 0,8902 256,54966828

Source 19,1445 126,5872 0,2146

Result 19,1445 126,5872 0,2146 196,20484182

Source 0,5480 154,4303 0,3179

Result 0,5480 154,4303 0,3179 400,66201971

Source 23,8511 199,5966 0,7253

Result 23,8511 199,5966 0,7253 2086,65888854

Source 66,9089 117,0985 0,6661

Result 66,9089 117,0985 0,6661 100,48820494

Source 60,4404 156,6462 0,1646

Result 60,4404 156,6462 0,1646 260,30396449

Source 79,0747 117,8506 0,8874

Result 79,0747 117,8506 0,8874 104,98231437

Source 34,5988 175,6259 0,8120

Result 34,5988 175,6259 0,8120 313,37046707

Source 81,9099 161,2202 0,3688

Result 81,9099 161,2202 0,3688 294,20657975

Source 20,9263 173,4608 0,8287

Result 20,9263 173,4608 0,8287 1428,80709808

Source 53,5908 112,0889 0,3537

Result 53,5908 112,0889 0,3537 267,87560448

Source 34,1050 170,4661 0,3134

Result 34,1050 170,4661 0,3134 368,17817332

Source 32,6608 198,4713 0,1785

Result 32,6608 198,4713 0,1785 526,11544334

Source 18,1893 148,0671 0,5476

Result 18,1893 148,0671 0,5476 324,84737500

Source 54,0279 143,7988 0,6376

Result 54,0279 143,7988 0,6376 221,34251668

Source 91,9075 178,4008 0,8724

Result 91,9075 178,4008 0,8724 206,39123177

Source 44,2936 119,7182 0,0446

Result 44,2936 119,7182 0,0446 146,69914089

Source 12,8665 129,3143 0,8882

Result 12,8665 129,3143 0,8882 666,32576943

Source 51,8058 135,4179 0,8059

Result 51,8058 135,4179 0,8059 164,64515933

Source 48,5495 173,2745 0,9188

Result 48,5495 173,2745 0,9188 775,11423949

Source 11,8238 104,5014 0,5732

Result 11,8238 104,5014 0,5732 618,33530023

Source 66,9974 153,7072 0,9305

Result 66,9974 153,7072 0,9305 259,08018033

Source 72,1890 155,4199 0,2898

Result 72,1890 155,4199 0,2898 174,34714856

Source 85,0650 187,2790 0,3493

Result 85,0650 187,2790 0,3493 253,30979433

Source 29,8892 176,4824 0,7004

Result 29,8892 176,4824 0,7004 338,74723743

Source 37,1422 105,6383 0,4652

Result 37,1422 105,6383 0,4652 189,41804509

Source 53,9088 111,4574 0,1371

Result 53,9088 111,4574 0,1371 233,46503950

Source 52,2305 165,1319 0,3675

Result 52,2305 165,1319 0,3675 550,05650981

Source 94,7552 146,7820 0,6503

Result 94,7552 146,7820 0,6503 133,06064257

Source 52,2661 178,0111 0,6008

Result 52,2661 178,0111 0,6008 300,79551883

Source 6,0512 156,3952 0,8007

Result 6,0512 156,3952 0,8007 349,29139037

Source 9,5807 143,6448 0,9837

Result 9,5807 143,6448 0,9837 423,72709627

Source 28,5856 195,2089 0,6777

Result 28,5856 195,2089 0,6777 501,89504070

Source 26,3882 183,2539 0,2817

Result 26,3882 183,2539 0,2817 973,64511065

Source 12,2104 157,2344 0,9619

Result 12,2104 157,2344 0,9619 683,26521812

Source 24,1917 106,6826 0,6754

Result 24,1917 106,6826 0,6754 162,44870779

Source 75,9783 139,3787 0,0971

Result 75,9783 139,3787 0,0971 160,82009320

Source 55,9972 164,7241 0,2156

Result 55,9972 164,7241 0,2156 334,24686473

Source 31,9076 176,9265 0,8888

Result 31,9076 176,9265 0,8888 799,35461533

Source 79,2287 165,5130 0,7071

Result 79,2287 165,5130 0,7071 232,85800112

Source 22,7680 141,7241 0,0958

Result 22,7680 141,7241 0,0958 320,19385680

Source 52,2318 158,5035 0,7114

Result 52,2318 158,5035 0,7114 215,05754065

Source 49,2761 119,0793 0,8299

Result 49,2761 119,0793 0,8299 165,10848949

Source 42,2595 114,5468 0,6112

Result 42,2595 114,5468 0,6112 164,06750976

Source 87,0927 104,3954 0,8326

Result 87,0927 104,3954 0,8326 46,98656283

Source 57,8275 155,5838 0,9274

Result 57,8275 155,5838 0,9274 998,45238733

Source 19,0111 185,5521 0,9425

Result 19,0111 185,5521 0,9425 360,97939228

Source 43,3338 112,4693 0,3289

Result 43,3338 112,4693 0,3289 218,98565594

Source 32,7278 102,0514 0,3218

Result 32,7278 102,0514 0,3218 198,16136465

Source 98,8283 114,4428 0,0297

Result 98,8283 114,4428 0,0297 31,80584771

Source 59,5863 196,9424 0,2571

Result 59,5863 196,9424 0,2571 339,31788391

Source 12,8280 138,7616 0,5104

Result 12,8280 138,7616 0,5104 240,88198798

Source 54,6842 102,5414 0,6020

Result 54,6842 102,5414 0,6020 122,85790340

Source 21,0644 174,0116 0,7602

Result 21,0644 174,0116 0,7602 396,31481999

Source 43,7348 125,9630 0,1763

Result 43,7348 125,9630 0,1763 164,56141413

Source 29,0560 175,6127 0,4762

Result 29,0560 175,6127 0,4762 375,79118783

Source 87,0729 125,4775 0,6903

Result 87,0729 125,4775 0,6903 686,53394738

Source 61,3958 117,0319 0,9154

Result 61,3958 117,0319 0,9154 135,99047483

Source 36,1967 193,7099 0,0088

Result 36,1967 193,7099 0,0088 327,53609475

Source 25,6447 165,2227 0,3381

Result 25,6447 165,2227 0,3381 578,89465676

Source 83,2351 188,6975 0,8217

Result 83,2351 188,6975 0,8217 259,61538044

Source 3,7208 133,8042 0,2353

Result 3,7208 133,8042 0,2353 1032,72980883

Source 28,7320 178,8201 0,0389

Result 28,7320 178,8201 0,0389 379,75122552

Source 8,6762 192,4961 0,7809

Result 8,6762 192,4961 0,7809 600,77717540

Source 18,1378 192,6189 0,4733

Result 18,1378 192,6189 0,4733 362,51693650

Source 69,7508 136,3266 0,2016

Result 69,7508 136,3266 0,2016 139,85677340

Source 22,3986 189,5209 0,6312

Result 22,3986 189,5209 0,6312 362,98344807

Source 33,5950 143,1259 0,0032

Result 33,5950 143,1259 0,0032 351,76572009

Source 63,0346 192,4825 0,6741

Result 63,0346 192,4825 0,6741 369,22751288

Source 94,5504 172,8244 0,2475

Result 94,5504 172,8244 0,2475 740,75116717

Source 7,0967 148,5695 0,8700

Result 7,0967 148,5695 0,8700 267,04347957

Source 91,8809 187,0683 0,3194

Result 91,8809 187,0683 0,3194 265,46682958

Source 82,3935 108,4881 0,0996

Result 82,3935 108,4881 0,0996 80,34119157

Source 36,8430 192,6700 0,8763

Result 36,8430 192,6700 0,8763 523,34182667

Source 88,3785 141,0832 0,2343

Result 88,3785 141,0832 0,2343 1472,38676481

Source 57,4877 137,4082 0,9212

Result 57,4877 137,4082 0,9212 216,77155654

Source 3,0884 123,8194 0,4519

Result 3,0884 123,8194 0,4519 426,06768351

Source 74,0826 111,6935 0,1909

Result 74,0826 111,6935 0,1909 100,11350617

Source 11,5232 175,2600 0,8076

Result 11,5232 175,2600 0,8076 335,13114302

Source 43,0474 134,6150 0,3000

Result 43,0474 134,6150 0,3000 222,49671017

Source 5,8581 120,5603 0,6247

Result 5,8581 120,5603 0,6247 8093,71540391

Source 32,2905 127,8840 0,3336

Result 32,2905 127,8840 0,3336 257,13438647

Source 32,8786 105,9902 0,5234

Result 32,8786 105,9902 0,5234 154,73066976

Source 15,8401 106,1894 0,8284

Result 15,8401 106,1894 0,8284 228,65872711

Source 89,0294 103,5018 0,1533

Result 89,0294 103,5018 0,1533 73,67858050

Source 0,6582 137,4817 0,3351

Result 0,6582 137,4817 0,3351 241,08361768

Source 88,4668 171,3585 0,9452

Result 88,4668 171,3585 0,9452 1000,22786077

Source 23,5294 151,7004 0,0024

Result 23,5294 151,7004 0,0024 400,48266343

Source 54,5068 161,8394 0,4504

Result 54,5068 161,8394 0,4504 705,74938802

Integrator завершен**Задание 4**

```
package functions.threads;
import functions.basic.Log;
import java.util.Random;
import java.util.concurrent.Semaphore;

public class Generator extends Thread {
    private Task task;
    private Semaphore semaphore;
    private Random random = new Random();

    public Generator(Task task, Semaphore semaphore) {
        this.task = task;
        this.semaphore = semaphore;
    }

    @Override
    public void run() {
        try {
            for (int i = 0; i < task.getTasks(); i++) {
                if (Thread.interrupted()) {
                    System.out.println("Generator interrupted");
                    return;
                }

                double base = 1 + random.nextDouble() * 9;
                double leftBorder = random.nextDouble() * 100;
                double rightBorder = 100 + random.nextDouble() * 100;
                double step = random.nextDouble();

                semaphore.acquire();
                try {
```

```
                task.setF(new Log(base));
                task.setLeft(leftBorder);
                task.setRight(rightBorder);
                task.setStep(step);
            } finally {
                semaphore.release();
            }

            System.out.printf("Source %.4f %.4f %.4f%n", leftBorder, rightBorder,
step);

            Thread.sleep(15);
        }
    } catch (InterruptedException e) {
        System.out.println("Generator was interrupted");
    }
  }
}
```

```
package functions.threads;
import functions.Function;
import java.util.concurrent.Semaphore;
import static functions.Functions.integrate;

public class Integrator extends Thread {
    private Task task;
    private Semaphore semaphore;

    public Integrator(Task task, Semaphore semaphore) {
        this.task = task;
        this.semaphore = semaphore;
    }
    @Override
    public void run() {
        int processed = 0;

        while (processed < task.getTasks() && !isInterrupted()) {
            Function function = null;
            double left = 0;
            double right = 0;
            double step = 0;

            try {
```

```
            semaphore.acquire();
            try {
                function = task.getF();
                left = task.getLeft();
                right = task.getRight();
                step = task.getStep();
                task.setF(null);
            } finally {
                semaphore.release();
            }
        } catch (InterruptedException e) {
            System.out.println("Integrator: прерван при ожидании семафора");
            Thread.currentThread().interrupt();
            return;
        }

        if (function != null) {
            processed++;
            try {
                double result = integrate(function, left, right, step);
                System.out.printf("Result %.4f %.4f %.4f %.8f%n", left, right, step,
result);
            } catch (Exception e) {
                System.out.println("Integrator error: " + e.getMessage());
            }
        }
        else {
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                System.out.println("Integrator прерван во время сна");
                Thread.currentThread().interrupt();
                return;
            }
        }
    }
}
```

```
public static void complicatedThreads() {
    System.out.println("Запуск улучшенной версии с семафором...");

    Task task = new Task(100);
    Semaphore semaphore = new Semaphore();
```

```
    // Создание потоков
    Generator generator = new Generator(task, semaphore);
    Integrator integrator = new Integrator(task, semaphore);

    // Установка приоритетов
    generator.setPriority(Thread.NORM_PRIORITY);
    integrator.setPriority(Thread.NORM_PRIORITY);

    // Запуск потоков
    generator.start();
    integrator.start();

    // Ожидание завершения потоков
    try {
        generator.join();
        integrator.join();
    } catch (InterruptedException e) {
        System.out.println("Main thread interrupted while waiting");
    }

    System.out.println("Улучшенная версия завершена");
}
```

3. УЛУЧШЕННАЯ ВЕРСИЯ С СЕМАФОРОМ

Запуск улучшенной версии с семафором...

Source 29,0968 139,7943 0,9950

Result 29,0968 139,7943 0,9950 333,19997494

Source 57,6392 110,0319 0,7288

Result 57,6392 110,0319 0,7288 155,14058205

Source 90,9771 100,3046 0,8622

Result 90,9771 100,3046 0,8622 23,95646234

Source 19,2779 115,9734 0,7930

Result 19,2779 115,9734 0,7930 326,88751149

Source 25,0799 142,5101 0,9645

Result 25,0799 142,5101 0,9645 233,92356834

Source 82,6450 138,5478 0,3598

Result 82,6450 138,5478 0,3598 166,84158953

Source 48,7714 171,2019 0,4906

Result 48,7714 171,2019 0,4906 311,35792991

Source 61,8182 135,3388 0,5959

Result 61,8182 135,3388 0,5959 237,01506160

Source 38,4469 187,6307 0,8852

Result 38,4469 187,6307 0,8852 1225,29295680

Source 33,4625 143,5190 0,3060

Result 33,4625 143,5190 0,3060 257,48677552

Source 7,1444 130,6203 0,5008

Result 7,1444 130,6203 0,5008 368,23157505

Source 26,5809 165,6434 0,3276

Result 26,5809 165,6434 0,3276 504,79360611

Source 12,7802 123,5453 0,5630

Result 12,7802 123,5453 0,5630 1337,06589435

Source 33,7068 197,4409 0,1790

Result 33,7068 197,4409 0,1790 424,37416041

Source 34,8958 105,6901 0,8936

Result 34,8958 105,6901 0,8936 247,65395249

Source 93,9290 188,6203 0,5289

Result 93,9290 188,6203 0,5289 245,84765286

Source 82,8079 151,5054 0,0809

Result 82,8079 151,5054 0,0809 307,72919657

Source 23,1390 152,7047 0,3033

Result 23,1390 152,7047 0,3033 309,22645466

Source 23,0513 106,2758 0,0392

Result 23,0513 106,2758 0,0392 168,72465332

Source 91,8923 166,9126 0,4956

Result 91,8923 166,9126 0,4956 174,89410148

Source 39,0061 191,7167 0,1492

Result 39,0061 191,7167 0,1492 333,37087044

Source 76,2734 128,0986 0,9971

Result 76,2734 128,0986 0,9971 130,01782007

Source 60,5984 160,0135 0,4200

Result 60,5984 160,0135 0,4200 211,00045455

Source 32,1561 155,7711 0,0140

Result 32,1561 155,7711 0,0140 261,67064866

Source 26,9675 199,8214 0,3649

Result 26,9675 199,8214 0,3649 392,69920577

Source 68,2490 158,2015 0,8872

Result 68,2490 158,2015 0,8872 284,05155767

Source 69,9872 122,7387 0,5323

Result 69,9872 122,7387 0,5323 143,46509995

Source 18,3167 102,6419 0,3815

Result 18,3167 102,6419 0,3815 229,03305723

Source 31,5351 183,7717 0,8819

Result 31,5351 183,7717 0,8819 1445,76342261

Source 37,9680 155,6655 0,7600

Result 37,9680 155,6655 0,7600 886,39493517

Source 52,3974 114,3048 0,4496

Result 52,3974 114,3048 0,4496 140,25669544

Source 76,4015 184,6456 0,2119

Result 76,4015 184,6456 0,2119 238,55562265

Source 95,7021 171,0932 0,9909

Result 95,7021 171,0932 0,9909 222,54345029

Source 98,4987 136,3902 0,4471

Result 98,4987 136,3902 0,4471 196,74541235

Source 92,7703 129,7922 0,5447

Result 92,7703 129,7922 0,5447 92,15525378

Source 57,6558 133,9458 0,5270

Result 57,6558 133,9458 0,5270 221,38437539

Source 24,2764 176,5397 0,1097

Result 24,2764 176,5397 0,1097 711,18762712

Source 44,7448 157,5874 0,2461

Result 44,7448 157,5874 0,2461 323,61077621

Source 74,6072 176,1079 0,8728

Result 74,6072 176,1079 0,8728 337,69784102

Source 63,5943 101,0097 0,6317

Result 63,5943 101,0097 0,6317 68330,31659474

Source 22,3120 120,1381 0,6837

Result 22,3120 120,1381 0,6837 218,13436740

Source 8,3803 118,7360 0,3771

Result 8,3803 118,7360 0,3771 219,15540340

Source 54,5326 183,9763 0,7144

Result 54,5326 183,9763 0,7144 804,92059269

Source 43,3188 105,7584 0,2759

Result 43,3188 105,7584 0,2759 132,99671731

Source 18,3053 145,6238 0,4967

Result 18,3053 145,6238 0,4967 253,52498135

Source 10,0493 175,4281 0,9733

Result 10,0493 175,4281 0,9733 416,88783278

Source 46,6239 145,3195 0,7786

Result 46,6239 145,3195 0,7786 633,10728177

Source 13,1012 126,6078 0,6004

Result 13,1012 126,6078 0,6004 641,60888081

Source 22,1567 189,2676 0,8910

Result 22,1567 189,2676 0,8910 420,55154884

Source 58,8325 116,1052 0,1083

Result 58,8325 116,1052 0,1083 204,36072264

Source 66,5056 126,0295 0,5662

Result 66,5056 126,0295 0,5662 129,79790256

Source 48,2325 177,4006 0,3687

Result 48,2325 177,4006 0,3687 377,81038457

Source 67,8990 149,0920 0,3990

Result 67,8990 149,0920 0,3990 167,98442751

Source 48,9784 151,6856 0,3567

Result 48,9784 151,6856 0,3567 232,99353480

Source 85,2103 104,1302 0,3933

Result 85,2103 104,1302 0,3933 54,93190065

Source 97,1368 122,2286 0,9831

Result 97,1368 122,2286 0,9831 148,32293415

Source 91,1900 104,4791 0,2837

Result 91,1900 104,4791 0,2837 26,47417621

Source 3,9005 168,0817 0,3914

Result 3,9005 168,0817 0,3914 348,15632317

Source 42,8046 183,3302 0,5751

Result 42,8046 183,3302 0,5751 455,12699816

Source 67,7516 178,1458 0,7872

Result 67,7516 178,1458 0,7872 432,33190507

Source 91,5716 136,1686 0,4904

Result 91,5716 136,1686 0,4904 100,65361466

Source 17,4404 191,6086 0,1153

Result 17,4404 191,6086 0,1153 801,88974294

Source 71,7810 131,9663 0,2692

Result 71,7810 131,9663 0,2692 182,27010276

Source 36,1732 155,7718 0,1522

Result 36,1732 155,7718 0,1522 653,12526607

Source 29,4948 196,6469 0,5074

Result 29,4948 196,6469 0,5074 751,29794783

Source 42,3960 117,2829 0,8284

Result 42,3960 117,2829 0,8284 157,58706735

Source 80,2149 197,3378 0,4800

Result 80,2149 197,3378 0,4800 255,14206701

Source 27,5684 187,7866 0,8494

Result 27,5684 187,7866 0,8494 341,71126890

Source 8,1091 180,9259 0,6382

Result 8,1091 180,9259 0,6382 429,29077596

Source 76,3155 150,0126 0,9297

Result 76,3155 150,0126 0,9297 301,24310478

Source 44,6877 145,4309 0,7226

Result 44,6877 145,4309 0,7226 355,22034168

Source 19,3507 195,3147 0,2074

Result 19,3507 195,3147 0,2074 349,84004306

Source 89,8189 147,7138 0,1829

Result 89,8189 147,7138 0,1829 147,22766513

Source 94,5574 108,6768 0,6551

Result 94,5574 108,6768 0,6551 31,22035400

Source 62,9759 176,0344 0,8470

Result 62,9759 176,0344 0,8470 284,35809116

Source 16,6234 142,2013 0,5543

Result 16,6234 142,2013 0,5543 359,66815384

Source 97,6047 140,9752 0,0297

Result 97,6047 140,9752 0,0297 95,85964165

Source 80,6156 143,0968 0,3492

Result 80,6156 143,0968 0,3492 332,24918404

Source 16,5044 163,2454 0,5227

Result 16,5044 163,2454 0,5227 583,32130594

Source 49,8876 127,1781 0,4456

Result 49,8876 127,1781 0,4456 149,76656381

Source 51,1098 161,8826 0,1575

Result 51,1098 161,8826 0,1575 261,05327472

Source 3,7103 113,8976 0,7734

Result 3,7103 113,8976 0,7734 252,97322205

Source 92,5671 171,3661 0,8829

Result 92,5671 171,3661 0,8829 278,69934845

Source 45,0156 163,9202 0,1765

Result 45,0156 163,9202 0,1765 283,48238175

Source 87,4936 141,1584 0,5476

Result 87,4936 141,1584 0,5476 111,82621058

Source 93,6446 185,2600 0,9486

Result 93,6446 185,2600 0,9486 624,02927799

Source 16,3121 196,1909 0,8298

Result 16,3121 196,1909 0,8298 729,64111904

Source 15,4906 198,9932 0,0802

Result 15,4906 198,9932 0,0802 1026,13339377

Source 17,0496 161,8074 0,0060

Result 17,0496 161,8074 0,0060 286,07746381

Source 62,9270 180,5630 0,0332

Result 62,9270 180,5630 0,0332 1289,93497624

Source 42,4928 181,6843 0,4988

Result 42,4928 181,6843 0,4988 281,12009606

Source 26,9118 117,6918 0,1106

Result 26,9118 117,6918 0,1106 633,53134544

Source 63,5890 182,2397 0,3962

Result 63,5890 182,2397 0,3962 276,80841085

Source 26,1587 150,9174 0,7361

Result 26,1587 150,9174 0,7361 340,97936335

Source 56,9846 162,0496 0,3840

Result 56,9846 162,0496 0,3840 291,06294009

Source 83,7648 187,7677 0,2110

Result 83,7648 187,7677 0,2110 487,87304783

Source 39,5907 109,7850 0,7815

Result 39,5907 109,7850 0,7815 358,61395008

Source 66,6874 106,5898 0,7651

Result 66,6874 106,5898 0,7651 97,73079725

Source 38,0674 179,1874 0,6233

Result 38,0674 179,1874 0,6233 289,64240085

Source 88,4579 191,0496 0,9668

Result 88,4579 191,0496 0,9668 437,02868435

Улучшенная версия завершена