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;
}

public synchronized TaskData getTaskData() {
    return new TaskData(f, left, right, step);
}
```

```
    public static class TaskData {
        public final Function function;
        public final double leftBorder;
        public final double rightBorder;
        public final double integrationStep;

        public TaskData(Function function, double leftBorder,
                        double rightBorder, double integrationStep) {
            this.function = function;
            this.leftBorder = leftBorder;
            this.rightBorder = rightBorder;
            this.integrationStep = integrationStep;
        }
    }
}
```

```
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.Functions;

public class SimpleIntegrator implements Runnable {
  private Task task;

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

  @Override
  public void run() {
    try {
      for (int i = 0; i < task.getTasks(); i++) {
        Task.TaskData data;

        synchronized (task) {
          data = task.getTaskData();
        }

        if (data.function == null) {
```

```
                Thread.sleep(1);
                continue;
            }

            double result = Functions.integrate(
                    data.function,
                    data.leftBorder,
                    data.rightBorder,
                    data.integrationStep
            );

            System.out.printf("Result %.4f %.4f %.4f %.8f%n", data.leftBorder,
data.rightBorder, data.integrationStep, result);
            Thread.sleep(10);
          }
       } catch (InterruptedException e) {
          System.out.println("Integrator error: " + e.getMessage());
       }
    }
}
```

```
private static void simpleThreads(){
    System.out.println("Метод реализующий потоковую версию программы");
    Task task = new Task(100);
    Thread generatorThread = new Thread(new SimpleGenerator(task));
    Thread integratorThread = new Thread(new SimpleIntegrator(task));
    generatorThread.start();
    integratorThread.start();
    try {
        generatorThread.join();
        integratorThread.join();
    } catch (InterruptedException e) {
        System.out.println("Main thread interrupted while waiting");
    }
}
```

2. ПРОСТАЯ МНОГОПОТОЧНАЯ ВЕРСИЯ

Метод реализующий потоковую версию программы

Source 66,7804 120,9304 0,0976

Result 66,7804 120,9304 0,0976 115,83129364

Source 21,3357 161,8128 0,0959

Result 21,3357 161,8128 0,0959 310,26861763

Source 43,7366 131,0702 0,1132

Result 43,7366 131,0702 0,1132 231,03647378

Source 12,5780 145,9634 0,4371

Result 43,7366 131,0702 0,1132 231,03647378

Source 3,3845 170,5577 0,4720

Result 3,3845 170,5577 0,4720 702,03038267

Source 5,1047 193,7542 0,9733

Result 5,1047 193,7542 0,9733 925,40652052

Source 29,5352 161,5331 0,0184

Result 29,5352 161,5331 0,0184 316,95435665

Source 26,6145 155,1842 0,5313

Result 26,6145 155,1842 0,5313 261,92682673

Source 44,4102 174,5846 0,3302

Result 44,4102 174,5846 0,3302 263,25953336

Source 1,6228 180,9956 0,3422

Result 44,4102 174,5846 0,3302 263,25953336

Source 72,5623 177,3834 0,8468

Result 1,6228 180,9956 0,3422 818,11804222

Result 72,5623 177,3834 0,8468 229,34695692

Source 19,8042 155,9313 0,3509

Source 48,1128 122,1927 0,9700

Result 19,8042 155,9313 0,3509 711,13114956

Source 94,7224 113,2487 0,0010

Result 48,1128 122,1927 0,9700 163,19175249

Source 15,6963 163,2647 0,7048

Result 94,7224 113,2487 0,0010 57,51878126

Source 75,8374 134,0044 0,1857

Result 75,8374 134,0044 0,1857 127,22076313

Source 0,5030 120,1424 0,6566

Result 0,5030 120,1424 0,6566 285,34789651

Source 1,6806 159,6271 0,3066

Result 1,6806 159,6271 0,3066 585,98377550

Source 47,8097 127,8823 0,4503

Result 47,8097 127,8823 0,4503 159,34635539

Source 7,0895 171,2933 0,6330

Result 7,0895 171,2933 0,6330 387,19900457

Source 57,3904 140,2118 0,8881

Result 57,3904 140,2118 0,8881 240,50485210

Source 22,3915 114,6210 0,4003

Result 22,3915 114,6210 0,4003 313,65354520

Source 63,1958 176,6388 0,4133

Result 63,1958 176,6388 0,4133 406,35703956

Source 27,4272 173,1984 0,3420

Result 27,4272 173,1984 0,3420 294,82168002

Source 33,6938 129,5277 0,6948

Result 33,6938 129,5277 0,6948 297,86697166

Source 48,3055 119,4622 0,5947

Result 48,3055 119,4622 0,5947 253,47246262

Source 93,2613 172,1589 0,4004

Result 93,2613 172,1589 0,4004 197,86226432

Source 9,7792 196,4411 0,1680

Result 9,7792 196,4411 0,1680 684,58136068

Source 90,0163 133,7858 0,4742

Result 90,0163 133,7858 0,4742 116,47842984

Source 81,1331 108,8807 0,4210

Result 81,1331 108,8807 0,4210 75,81636447

Source 60,4844 132,7061 0,4849

Result 60,4844 132,7061 0,4849 441,45782696

Source 78,4740 127,2650 0,2367

Result 78,4740 127,2650 0,2367 139,88370920

Source 66,4861 183,3625 0,8614

Result 66,4861 183,3625 0,8614 249,24461079

Source 11,3149 133,2507 0,8354

Result 11,3149 133,2507 0,8354 291,60528912

Source 4,6389 118,4889 0,0822

Result 4,6389 118,4889 0,0822 202,04704237

Source 86,3420 160,0283 0,4191

Result 86,3420 160,0283 0,4191 316,68710613

Result 86,3420 160,0283 0,4191 316,68710613

Source 13,8631 192,3167 0,1344

Source 6,2571 106,1283 0,0870

Result 13,8631 192,3167 0,1344 362,75686855

Source 31,2828 155,8586 0,8100

Result 6,2571 106,1283 0,0870 216,68159029

Source 26,5841 126,1856 0,7161

Result 31,2828 155,8586 0,8100 359,80526778

Source 71,9114 120,2829 0,6543

Result 26,5841 126,1856 0,7161 418,83603302

Source 58,3324 127,1791 0,7447

Result 71,9114 120,2829 0,6543 111,78883709

Source 32,5141 149,3972 0,6340

Result 58,3324 127,1791 0,7447 500,28133159

Source 29,4055 123,6618 0,0537

Result 32,5141 149,3972 0,6340 418,44991882

Source 45,0366 147,2540 0,8411

Result 29,4055 123,6618 0,0537 212,97133830

Source 42,6471 130,4968 0,3848

Result 42,6471 130,4968 0,3848 712,39721050

Source 29,9910 131,8960 0,7984

Result 29,9910 131,8960 0,7984 250,73410212

Source 93,4151 196,6155 0,9879

Result 93,4151 196,6155 0,9879 245,18366065

Source 25,1897 111,2557 0,1844

Result 25,1897 111,2557 0,1844 196,85271944

Source 34,5729 108,3122 0,4390

Result 34,5729 108,3122 0,4390 183,70336006

Source 90,2182 102,2251 0,0420

Result 90,2182 102,2251 0,0420 26,62880541

Source 1,7909 115,5933 0,8200

Result 1,7909 115,5933 0,8200 451,28719246

Source 12,0361 158,9110 0,7895

Result 12,0361 158,9110 0,7895 599,48329364

Source 37,0456 160,0930 0,7018

Result 37,0456 160,0930 0,7018 408,74052000

Source 62,0990 101,0926 0,5989

Result 62,0990 101,0926 0,5989 104,83030883

Source 73,2107 108,0752 0,6220

Result 73,2107 108,0752 0,6220 138,30997976

Source 80,3900 158,8951 0,9583

Result 80,3900 158,8951 0,9583 185,60592829

Source 67,6278 117,0341 0,9987

Result 67,6278 117,0341 0,9987 282,79420920

Source 86,7525 174,6962 0,4850

Result 86,7525 174,6962 0,4850 342,67310577

Source 97,5486 189,4103 0,6722

Result 97,5486 189,4103 0,6722 204,67294706

Source 44,9149 128,2815 0,3055

Result 44,9149 128,2815 0,3055 413,55035846

Source 55,3391 191,9386 0,5147

Result 55,3391 191,9386 0,5147 299,62101652

Source 29,5969 176,2631 0,5690

Result 29,5969 176,2631 0,5690 376,11912088

Source 6,3367 144,1424 0,8012

Result 6,3367 144,1424 0,8012 270,72742066

Source 72,0896 121,7784 0,9102

Result 72,0896 121,7784 0,9102 180,93146510

Source 97,2501 104,4897 0,2818

Result 97,2501 104,4897 0,2818 26,34946774

Source 84,6983 155,4830 0,9290

Result 84,6983 155,4830 0,9290 180,36436923

Source 27,9931 107,7279 0,1472

Result 27,9931 107,7279 0,1472 740,02851001

Source 89,4399 133,8558 0,0023

Result 89,4399 133,8558 0,0023 2675,47543273

Source 24,8434 131,9122 0,3283

Result 24,8434 131,9122 0,3283 233,33724996

Source 1,3849 185,8294 0,9944

Result 1,3849 185,8294 0,9944 439,46409268

Source 77,8152 198,6426 0,8226

Result 77,8152 198,6426 0,8226 257,94605744

Source 42,0352 163,7631 0,6842

Result 42,0352 163,7631 0,6842 337,09192477

Source 30,1395 135,0263 0,9869

Result 30,1395 135,0263 0,9869 459,74471299

Source 83,1272 179,7733 0,9723

Result 83,1272 179,7733 0,9723 251,70882957

Source 32,7384 150,8802 0,9899

Result 32,7384 150,8802 0,9899 407,66384866

Source 46,2522 193,4375 0,1879

Result 46,2522 193,4375 0,1879 3043,68096677

Source 1,5886 109,6902 0,8638

Result 1,5886 109,6902 0,8638 275,78251918

Source 8,6835 141,8867 0,0595

Result 8,6835 141,8867 0,0595 239,84630028

Source 40,8606 196,8411 0,8483

Result 40,8606 196,8411 0,8483 320,61377942

Source 90,8024 128,3029 0,1667

Result 90,8024 128,3029 0,1667 77,11739318

Source 19,7278 113,6349 0,9673

Result 19,7278 113,6349 0,9673 238,42514621

Source 25,5113 109,8366 0,9397

Result 25,5113 109,8366 0,9397 196,63016276

Source 15,7328 150,6115 0,0299

Result 15,7328 150,6115 0,0299 747,62643780

Source 35,0270 150,9596 0,5225

Result 35,0270 150,9596 0,5225 479,14736624

Source 62,6043 118,6884 0,4973

Result 62,6043 118,6884 0,4973 126,17743306

Source 65,4438 133,2702 0,0660

Result 65,4438 133,2702 0,0660 165,55638507

Source 65,4141 141,6340 0,9066

Result 65,4141 141,6340 0,9066 562,56927791

Source 54,4561 137,7644 0,2682

Result 54,4561 137,7644 0,2682 212,11196740

Source 70,9639 138,3915 0,7940

Result 70,9639 138,3915 0,7940 161,50970075

Source 32,6330 140,5904 0,2277

Result 32,6330 140,5904 0,2277 220,56831359

Source 6,2942 148,7678 0,3069

Result 6,2942 148,7678 0,3069 302,07615838

Source 83,6341 188,3013 0,9254

Result 83,6341 188,3013 0,9254 3233,04399310

Source 23,2050 181,0035 0,9481

Result 23,2050 181,0035 0,9481 514,01295052

Source 13,7574 143,6216 0,9035

Result 13,7574 143,6216 0,9035 516,49138436

Source 11,6793 184,7401 0,1427

Result 11,6793 184,7401 0,1427 387,53841510

Source 6,2106 117,7432 0,9563

Result 6,2106 117,7432 0,9563 205,24459144

Source 70,0989 177,8988 0,5694

Result 70,0989 177,8988 0,5694 403,51052287

Source 60,4957 130,1668 0,8989

Result 60,4957 130,1668 0,8989 140,75662325

Source 45,1455 180,9882 0,4468

Result 45,1455 180,9882 0,4468 361,77755101

Задание 4

```
package functions.threads;

public class Semaphore {
    private int readers = 0;
    private int writers = 0;
```

```
    private int writeRequests = 0;

    public synchronized void startRead() throws InterruptedException {
        while (writers > 0 || writeRequests > 0) {
            wait();
        }
        readers++;
    }

    public synchronized void endRead() {
        readers--;
        notifyAll();
    }

    public synchronized void startWrite() throws InterruptedException {
        writeRequests++;
        while (readers > 0 || writers > 0) {
            wait();
        }
        writeRequests--;
        writers++;
    }

    public synchronized void endWrite() {
        writers--;
        notifyAll();
    }
}
```

```
    }

    @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.startWrite();
                try {
                    task.setF(new Log(base));
                    task.setLeft(leftBorder);
                    task.setRight(rightBorder);
                    task.setStep(step);
                } finally {
                    semaphore.endWrite();
                }

                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.Functions;

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() {
        try {
            for (int i = 0; i < task.getTasks(); i++) {

                if (Thread.interrupted()) {
                    System.out.println("Integrator interrupted");
                    return;
                }
```

```
            Task.TaskData data;


            semaphore.startRead();
            try {
                data = task.getTaskData();
            } finally {
                semaphore.endRead();
            }


            if (data.function == null) {
                Thread.sleep(1);
                continue;
            }


            double result = Functions.integrate(
                    data.function,
                    data.leftBorder,
                    data.rightBorder,
                    data.integrationStep
            );


            System.out.printf("Result %.4f %.4f %.4f %.8f%n", data.leftBorder,
data.rightBorder, data.integrationStep, result);
            Thread.sleep(15);
          }
       } catch (InterruptedException e) {
          System.out.println("Integrator error: " + e.getMessage());
       }
    }
}
```

```
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 28,0531 169,3407 0,9101

Result 28,0531 169,3407 0,9101 448,03171925

Result 28,0531 169,3407 0,9101 448,03171925

Source 10,4783 128,3794 0,5576

Result 10,4783 128,3794 0,5576 266,89854897

Source 36,7934 157,5752 0,3591

Source 3,5860 167,1410 0,3004

Result 36,7934 157,5752 0,3591 251,67288110

Source 32,5368 158,8746 0,7615

Result 3,5860 167,1410 0,3004 392,53548715

Source 48,1313 145,2664 0,4280

Result 48,1313 145,2664 0,4280 209,30763949

Source 62,4408 110,8374 0,6413

Result 62,4408 110,8374 0,6413 148,96032689

Source 62,1437 141,2723 0,4817

Result 62,1437 141,2723 0,4817 183,52187452

Source 0,0830 144,4474 0,8437

Result 0,0830 144,4474 0,8437 389,80743534

Source 3,9139 114,1126 0,2545

Result 3,9139 114,1126 0,2545 239,42355111

Source 27,7712 141,7732 0,2102

Result 27,7712 141,7732 0,2102 278,85867949

Source 89,3905 165,2264 0,3616

Result 89,3905 165,2264 0,3616 198,49625976

Source 54,7492 153,2431 0,9672

Result 54,7492 153,2431 0,9672 207,75736053

Source 55,0416 165,0449 0,6495

Result 55,0416 165,0449 0,6495 1198,59861543

Source 46,6874 190,5128 0,6409

Result 46,6874 190,5128 0,6409 382,49217263

Source 54,0721 172,4729 0,3842

Result 54,0721 172,4729 0,3842 3117,16472607

Source 66,8495 135,9401 0,5673

Result 66,8495 135,9401 0,5673 140,81532773

Source 98,8881 100,8613 0,4118

Result 98,8881 100,8613 0,4118 13,73474811

Source 54,3260 134,6836 0,3129

Result 54,3260 134,6836 0,3129 456,06407535

Source 87,6969 108,9854 0,7308

Result 87,6969 108,9854 0,7308 94,97907201

Source 89,7250 160,4398 0,5721

Result 89,7250 160,4398 0,5721 153,62396127

Source 78,1357 120,5448 0,1689

Result 78,1357 120,5448 0,1689 98,41056482

Source 23,7107 170,8347 0,3587

Result 23,7107 170,8347 0,3587 468,04200031

Source 95,8182 100,7011 0,1053

Result 95,8182 100,7011 0,1053 10,10270392

Source 68,7804 193,4859 0,5563

Result 68,7804 193,4859 0,5563 384,84146968

Source 90,6898 143,1679 0,4298

Result 90,6898 143,1679 0,4298 120,74217251

Source 79,6711 108,7506 0,1467

Result 79,6711 108,7506 0,1467 5340,76116162

Source 79,0825 146,3701 0,1799

Result 79,0825 146,3701 0,1799 147,84243098

Source 72,7853 120,1742 0,6792

Result 72,7853 120,1742 0,6792 149,21988908

Source 34,3598 182,1872 0,5482

Result 34,3598 182,1872 0,5482 503,47053248

Source 60,0276 128,6572 0,8003

Result 60,0276 128,6572 0,8003 135,34686254

Source 14,0426 185,7960 0,2001

Result 14,0426 185,7960 0,2001 410,06054223

Source 97,4423 151,3533 0,9086

Result 97,4423 151,3533 0,9086 158,78851050

Source 36,3057 120,1296 0,0950

Result 36,3057 120,1296 0,0950 222,59425346

Source 74,0420 151,3172 0,2512

Result 74,0420 151,3172 0,2512 226,29159198

Source 43,7373 193,4668 0,2504

Result 43,7373 193,4668 0,2504 308,54154649

Source 91,2197 168,9153 0,0296

Result 91,2197 168,9153 0,0296 304,78821644

Source 17,2501 133,6684 0,1422

Result 17,2501 133,6684 0,1422 222,10557164

Source 8,1327 171,0167 0,3287

Result 8,1327 171,0167 0,3287 379,88255648

Source 28,0362 147,5763 0,6806

Result 28,0362 147,5763 0,6806 249,08059528

Source 53,4506 158,9783 0,9386

Result 53,4506 158,9783 0,9386 213,78195979

Source 40,4556 119,0901 0,9557

Result 40,4556 119,0901 0,9557 216,28441635

Source 28,8570 101,4421 0,3832

Result 28,8570 101,4421 0,3832 157,20923042

Source 3,7496 124,4370 0,1911

Result 3,7496 124,4370 0,1911 853,49758786

Source 54,6969 182,9579 0,7226

Result 54,6969 182,9579 0,7226 366,13622568

Source 95,4685 168,1519 0,5994

Result 95,4685 168,1519 0,5994 169,15107462

Source 33,1580 184,4482 0,9462

Result 33,1580 184,4482 0,9462 670,50357287

Source 4,5055 168,2819 0,0796

Result 4,5055 168,2819 0,0796 329,04841900

Source 29,3796 136,5310 0,0230

Result 29,3796 136,5310 0,0230 358,57067020

Source 66,6272 144,3304 0,9485

Result 66,6272 144,3304 0,9485 212,27704787

Source 95,8772 111,7159 0,9240

Result 95,8772 111,7159 0,9240 33,83169396

Source 90,3905 186,4589 0,1470

Result 90,3905 186,4589 0,1470 215,99256219

Source 90,6745 146,4889 0,1855

Result 90,6745 146,4889 0,1855 121,99146249

Source 36,3374 109,4595 0,6133

Result 36,3374 109,4595 0,6133 459,07156681

Source 64,5335 109,3018 0,8274

Result 64,5335 109,3018 0,8274 185,30962377

Result 64,5335 109,3018 0,8274 185,30962377

Source 37,3578 120,0147 0,7639

Source 17,6731 155,4819 0,6945

Result 17,6731 155,4819 0,6945 331,65981254

Result 17,6731 155,4819 0,6945 331,65981254

Source 73,1115 150,7608 0,2737

Source 51,3667 135,0773 0,5365

Result 51,3667 135,0773 0,5365 2371,90820046

Result 51,3667 135,0773 0,5365 2371,90820046

Source 36,3996 139,6692 0,5975

Result 36,3996 139,6692 0,5975 212,94178682

Source 64,8096 144,6540 0,3930

Result 64,8096 144,6540 0,3930 2567,19900836

Source 54,3751 158,2627 0,9229

Result 54,3751 158,2627 0,9229 406,30366036

Source 72,6473 139,0592 0,1005

Source 52,4861 117,1810 0,4979

Result 72,6473 139,0592 0,1005 137,53756100

Result 52,4861 117,1810 0,4979 130,11972472

Source 43,5538 144,1684 0,9873

Source 74,0380 102,6118 0,5220

Result 43,5538 144,1684 0,9873 295,89573532

Result 74,0380 102,6118 0,5220 62,20517301

Source 85,2523 136,6300 0,0727

Source 93,2696 122,1036 0,0452

Result 85,2523 136,6300 0,0727 186,33045289

Source 26,3685 181,6238 0,3373

Result 93,2696 122,1036 0,0452 59,58413319

Result 26,3685 181,6238 0,3373 324,82062789

Source 84,8311 163,2740 0,5641

Result 84,8311 163,2740 0,5641 184,18178439

Source 70,5056 144,6498 0,8805

Result 70,5056 144,6498 0,8805 213,50172105

Source 37,5545 193,9318 0,2569

Source 22,9340 118,3610 0,0315

Result 37,5545 193,9318 0,2569 654,54124716

Source 25,4095 157,1261 0,0496

Result 25,4095 157,1261 0,0496 333,05278557

Source 15,2847 130,4583 0,7307

Result 15,2847 130,4583 0,7307 485,23531051

Source 73,5323 139,6391 0,9544

Result 73,5323 139,6391 0,9544 148,22599626

Source 78,6371 123,6605 0,7475

Result 78,6371 123,6605 0,7475 98,64316626

Source 82,2205 117,4387 0,3704

Result 82,2205 117,4387 0,3704 119,14626195

Source 40,3929 123,7674 0,5504

Result 40,3929 123,7674 0,5504 181,08004955

Source 96,7833 106,6053 0,2909

Result 96,7833 106,6053 0,2909 20,65914518

Source 45,3259 136,8301 0,8976

Result 45,3259 136,8301 0,8976 236,02029074

Source 36,1377 173,2493 0,6051

Result 36,1377 173,2493 0,6051 320,47272642

Source 26,1974 141,4445 0,4620

Result 26,1974 141,4445 0,4620 249,45170233

Source 18,0428 180,9351 0,9389

Result 18,0428 180,9351 0,9389 390,55266117

Source 74,3309 167,1840 0,0594

Result 74,3309 167,1840 0,0594 192,74283879

Source 43,8602 179,1260 0,7240

Result 43,8602 179,1260 0,7240 592,46745935

Source 85,5323 175,1866 0,2944

Result 85,5323 175,1866 0,2944 194,29173375

Source 30,7340 161,1674 0,4377

Result 30,7340 161,1674 0,4377 266,34238537

Source 81,7510 181,9119 0,6917

Result 81,7510 181,9119 0,6917 213,09952105

Source 4,0782 145,3220 0,2046

Result 4,0782 145,3220 0,2046 377,41273780

Source 33,8680 192,0846 0,6287

Result 33,8680 192,0846 0,6287 333,31088366

Source 88,2653 157,3746 0,2015

Result 88,2653 157,3746 0,2015 880,20787916

Source 36,0054 131,8386 0,6355

Result 36,0054 131,8386 0,6355 191,76439816

Source 58,4034 118,3205 0,5568

Result 58,4034 118,3205 0,5568 131,28280795

Source 6,7455 198,0788 0,8795

Result 6,7455 198,0788 0,8795 1176,56602753

Source 41,0946 165,6535 0,6868

Result 41,0946 165,6535 0,6868 262,30553514

Source 17,6511 165,0751 0,9001

Result 17,6511 165,0751 0,9001 390,37794780

Source 77,0655 161,7705 0,2585

Result 77,0655 161,7705 0,2585 185,11858765

Source 74,4773 114,3048 0,9108

Result 74,4773 114,3048 0,9108 110,03648118

Source 83,5581 168,0059 0,2215

Result 83,5581 168,0059 0,2215 194,07268174

Улучшенная версия завершена