Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе №6

Дисциплина: «ООП»

Тема «Разработать приложение, формирующее в одном потоке вычислений набор заданий для интегрирования, а во втором потоке – вычисляющее значения интегралов функций.»

Выполнил: Зорин Дмитрий Сергеевич

Группа: 6201-120303D

Самара, 2025

Задание на лабораторную работу

Задание 1

В класс Functions добавил метод для вычисления интеграла функции методом трапеций

```
public static double integrate(Function f, double leftX, double rightX, double step) {  no usages
    if (leftX < f.getLeftDomainBorder() || rightX > f.getRightDomainBorder()) {
        throw new IllegalArgumentException("Интеграл выходит за область определения функции");
    }
    if (step <= 0) throw new IllegalArgumentException("Шаг должен быть > 0");

    double sum = 0;
    double x = leftX;

    while (x < rightX) {
        double nextX = Math.min(x + step, rightX);
        sum += (f.getFunctionValue(x) + f.getFunctionValue(nextX)) / 2 * (nextX - x);
        x = nextX;
    }

    return sum;
}
```

Метод делит интервал и суммирует площади трапеций
В методе Main добавил проверку работы метода интегрирования:

```
    System.out.println("Проверка интегрирования экспоненты на [0,1]");
    Function f = new Exp();
    double left = 0;
    double right = 1;
    double exact = Math.exp(1) - 1;
    double step = 0.1;
    double integral;

    do {
        integral = Functions.integrate(f, left, right, step);
        step /= 2;
    } while (Math.abs(integral - exact) > 1e-7);

    System.out.printf("Интеграл e^x на [0,1] = %.10f%n", integral);
    System.out.printf("шаг дискретизации = %.10f%n", step * 2);

    System.out.println("\nNon-threads");
    nonThread();

    System.out.println("\nSimple threads");
    simpleThreads();

    System.out.println("\nComplicated threads");
    complicatedThreads();
}
```

Задание 2

Во 2 задании я создал пакет Threads и описал класс Task:

```
package functions.threads;

import functions.Function;

public class Task {  7 usages

    private Function function;  2 usages
    private double leftBorder;  2 usages
    private double rightBorder;  2 usages
    private double step;  2 usages
    private int tasksCount;  2 usages
    private boolean ready = false;  2 usages

    public synchronized boolean isReady() { return ready; }  1 usage
    public synchronized void setReady(boolean ready) { this.ready = ready; }  2 usages

    public Function getFunction() { return function; }
    public void setFunction(Function function) { this.function = function; }

    public double getLeftBorder() { return leftBorder; }  1 usage
    public void setLeftBorder(double leftBorder) { this.leftBorder = leftBorder; }  2 usages

    public double getRightBorder() { return rightBorder; }  1 usage
    public void setRightBorder(double rightBorder) { this.rightBorder = rightBorder; }  2 usages

    public double getStep() { return step; }  1 usage
    public void setStep(double step) { this.step = step; }  2 usages

    public int getTasksCount() { return tasksCount; }  3 usages
    public void setTasksCount(int tasksCount) { this.tasksCount = tasksCount; }  1 usage
}
```

А также метод nonThread():

```java
public static void nonThread() {  1 usage   Dima *
    Task task = new Task();
    task.setTasksCount(100);
    Random rand = new Random();

    for (int i = 0; i < task.getTasksCount(); i++) {
        double base = 1 + 9 * rand.nextDouble();
        double left = 100 * rand.nextDouble();
        double right = 100 + 100 * rand.nextDouble();
        double step = rand.nextDouble();
        task.setFunction(new Log(base));
        task.setLeftBorder(left);
        task.setRightBorder(right);
        task.setStep(step);

        System.out.printf("Source %.5f %.5f %.5f%n", left, right, step);

        double integral = Functions.integrate(task.getFunction(), left, right, step);

        System.out.printf("Result %.5f %.5f %.5f %.10f%n", left, right, step, integral);
    }
}
```

Метод nonThread() показывает как последовательная обработка через методы объекта Taskc, а класс Functions позволяет интегрировать функции без многопоточности

Задание 3

Необходимо реализовать два потока: генератор задач и интегратор

Класс SimpleGenerator:

```java
package functions.threads;

import functions.Functions;
import functions.basic.Log;

import java.util.Random;

public class SimpleGenerator implements Runnable {  1 usage

    private Task task;  9 usages
    private Random rnd = new Random();  4 usages

    public SimpleGenerator(Task task) {  5 usages
        this.task = task;
    }

    @Override
    public void run() {
        for (int i = 0; i < task.getTasksCount(); i++) {
            double base = 1 + 9 * rnd.nextDouble();
            double left = rnd.nextDouble() * 100;
            double right = 100 + rnd.nextDouble() * 100;
            double step = rnd.nextDouble();

            synchronized (task) {
                task.setFunction(new Log(base));
                task.setLeftBorder(left);
                task.setRightBorder(right);
                task.setStep(step);
                task.setReady(true);
                task.notifyAll();
            }

            System.out.printf("Source %.5f %.5f %.5f%n", left, right, step);

            try { Thread.sleep( millis: 10); } catch (InterruptedException ignored) {}
```

run() — метод, который вызывается при старте потока

Класс SimpleIntegrator:

```java
package functions.threads;

import functions.Functions;

public class SimpleIntegrator implements Runnable {  1 usage

    private Task task;  10 usages

    public SimpleIntegrator(Task task) {  5 usages
        this.task = task;
    }

    @Override
    public void run() {
        for (int i = 0; i < task.getTasksCount(); i++) {
            synchronized (task) {
                while (!task.isReady()) {
                    try { task.wait(); } catch (InterruptedException ignored) {}
                }

                double left = task.getLeftBorder();
                double right = task.getRightBorder();
                double step = task.getStep();
                double result = Functions.integrate(task.getFunction(), left, right, step);

                System.out.printf("Result %.5f %.5f %.5f %.10f%n", left, right, step, result);

                task.setReady(false);
            }
        }
```

Задание 4

В задании 4 необходимо было использовать семафор для предотвращения пропуска задач интегратором

Класс семафора:

```java
package functions.threads;

public class Semaphore {  6 usages  Dima *
    private boolean available = false;   4 usages

    public synchronized void acquireWrite() throws InterruptedException {  1 usage  Dima *
        while (available) {
            wait();
        }
    }

    public synchronized void releaseWrite() {  1 usage  Dima *
        available = true;
        notifyAll();
    }

    public synchronized void acquireRead() throws InterruptedException {  1 usage  Dima *
        while (!available) {
            wait();
        }
    }

    public synchronized void releaseRead() {  1 usage  Dima *
        available = false;
        notifyAll();
    }
}
```

Семафор различает операции чтения и записи в защищаемый объект

Также необходимо создать 2 класса Generator и Integrator:

Примет 1 из классов:

```java
import functions.Functions;

public class Integrator extends Thread {  2 usages  Dima *
    private Task task;  6 usages
    private Semaphore semaphore;  3 usages

    public Integrator(Task task, Semaphore semaphore) {  5 usages  Dima
        this.task = task;
        this.semaphore = semaphore;
    }

    @Override  Dima *
    public void run() {
        for (int i = 0; i < task.getTasksCount(); i++) {
            if (Thread.currentThread().isInterrupted()) return;

            try {
                semaphore.acquireRead();

                Function f = task.getFunction();
                double left = task.getLeftBorder();
                double right = task.getRightBorder();
                double step = task.getStep();

                double result = Functions.integrate(f, left, right, step);

                System.out.printf("Result %.4f %.4f %.4f %.8f%n", left, right, step, result);

                semaphore.releaseRead();

                Thread.sleep( millis: 1);
            } catch (InterruptedException e) {
                return;
            } catch (IllegalArgumentException e) {
                System.out.println("Ошибка интегрирования: " + e.getMessage());
            }
```

В Main надо создать метод complicatedThreads(), который создаёт и запускает два потока Generator и Integrator и проверяет их работу

```
public static void complicatedThreads() {  1 usage   Dima *
    Task task = new Task();
    task.setTasksCount(100);

    Semaphore semaphore = new Semaphore();

    Generator generator = new Generator(task, semaphore);
    Integrator integrator = new Integrator(task, semaphore);

    generator.start();
    integrator.start();

    try {
        Thread.sleep( millis: 50);
        generator.interrupt();
        integrator.interrupt();

        generator.join();
        integrator.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

Пример работы Main:

Проверка интегрирования экспоненты на [0,1]
Интеграл e^x на [0,1] = 1,7182819159
шаг дискретизации = 0,0007812500

Non-threads
Source 55,17980 156,05576 0,79139
Result 55,17980 156,05576 0,79139 209,8731632687
Source 18,65254 157,59260 0,82393
Result 18,65254 157,59260 0,82393 701,8132011188
Source 90,30876 177,56582 0,44396
Result 90,30876 177,56582 0,44396 246,3417174709
Source 57,14839 148,91374 0,97736
Result 57,14839 148,91374 0,97736 221,2782532514
Source 88,73789 190,32815 0,27276
Result 88,73789 190,32815 0,27276 237,4679172450
Source 91,01301 109,28409 0,13326
Result 91,01301 109,28409 0,13326 44,4214391053
Source 3,50891 139,01596 0,41763
Result 3,50891 139,01596 0,41763 251,3984000155

Source 78,69250 178,11980 0,12807
Result 78,69250 178,11980 0,12807 319,0690966666
Source 43,88853 142,18416 0,06003
Result 43,88853 142,18416 0,06003 589,2339357579
Source 61,23197 155,69962 0,88000
Result 61,23197 155,69962 0,88000 825,7686094654
Source 69,14596 109,46996 0,81819
Result 69,14596 109,46996 0,81819 86,5235413127
Source 10,03575 183,84506 0,51583
Result 10,03575 183,84506 0,51583 413,8275735997
Source 23,20575 191,31905 0,95030
Result 23,20575 191,31905 0,95030 374,1666824959
Source 9,44794 123,38403 0,77658
Result 9,44794 123,38403 0,77658 492,4573665054
Source 54,31642 196,00308 0,86980
Result 54,31642 196,00308 0,86980 352,3867412649
Source 30,79434 175,66352 0,33090
Result 30,79434 175,66352 0,33090 327,0620715340
Source 88,87568 133,69400 0,72671
Result 88,87568 133,69400 0,72671 216,3437959817
Source 79,52605 120,37704 0,87524
Result 79,52605 120,37704 0,87524 85,3632222168
Source 53,55268 172,36087 0,05877
Result 53,55268 172,36087 0,05877 250,4641547114
Source 22,10559 166,97230 0,14949
Result 22,10559 166,97230 0,14949 380,9651762709
Source 64,84805 115,25493 0,76641
Result 64,84805 115,25493 0,76641 432,3959429799
Source 24,27599 141,90487 0,24725
Result 24,27599 141,90487 0,24725 254,6861753320
Source 30,23232 134,74829 0,91540
Result 30,23232 134,74829 0,91540 341,5582393255
Source 57,35367 159,00346 0,84935
Result 57,35367 159,00346 0,84935 406,8563938374
Source 30,00330 122,29206 0,12112
Result 30,00330 122,29206 0,12112 476,2589699406
Source 54,18550 145,21233 0,25983
Result 54,18550 145,21233 0,25983 292,5789614074
Source 74,00307 139,25954 0,51834
Result 74,00307 139,25954 0,51834 150,9187112295

Source 21,41092 118,97652 0,37299
Result 21,41092 118,97652 0,37299 328,8664934585
Source 52,33531 178,43995 0,57041
Result 52,33531 178,43995 0,57041 702,2311796905
Source 39,66952 143,53562 0,95067
Result 39,66952 143,53562 0,95067 226,5470254291
Source 24,22333 170,94852 0,22174
Result 24,22333 170,94852 0,22174 466,9460731406
Source 8,39484 102,39147 0,83850
Result 8,39484 102,39147 0,83850 222,6763033977
Source 14,14979 198,31833 0,14759
Result 14,14979 198,31833 0,14759 565,2682333286
Source 16,83496 169,21572 0,00011
Result 16,83496 169,21572 0,00011 370,2300624200
Source 9,41835 194,53985 0,10492
Result 9,41835 194,53985 0,10492 446,6466386212
Source 47,94059 169,46255 0,49815
Result 47,94059 169,46255 0,49815 262,5193741990
Source 78,27824 147,81341 0,63829
Result 78,27824 147,81341 0,63829 616,2588795509
Source 46,76110 155,95245 0,80106
Result 46,76110 155,95245 0,80106 227,3718191613
Source 52,75205 136,78058 0,64479
Result 52,75205 136,78058 0,64479 323,4752264386
Source 9,98932 145,69255 0,04451
Result 9,98932 145,69255 0,04451 319,1129409290
Source 45,30760 160,99990 0,55167
Result 45,30760 160,99990 0,55167 308,1867672931
Source 78,29561 190,19909 0,12642
Result 78,29561 190,19909 0,12642 316,7866664066
Source 93,51441 150,43252 0,11091
Result 93,51441 150,43252 0,11091 172,4611339675
Source 38,09188 176,58781 0,76434
Result 38,09188 176,58781 0,76434 342,5558601918
Source 51,82543 163,70364 0,06431
Result 51,82543 163,70364 0,06431 240,9307795148
Source 64,36618 181,69767 0,89432
Result 64,36618 181,69767 0,89432 365,5304171616
Source 74,62457 109,90509 0,18321
Result 74,62457 109,90509 0,18321 80,3334465774

Source 61,22838 126,51378 0,04002
Result 61,22838 126,51378 0,04002 448,3226171959
Source 59,68084 153,70484 0,43672
Result 59,68084 153,70484 0,43672 219,6877122005
Source 81,80782 103,15488 0,53093
Result 81,80782 103,15488 0,53093 47,1857685384
Source 46,14948 188,52900 0,29229
Result 46,14948 188,52900 0,29229 5474,8421399785
Source 34,14321 171,69127 0,69877
Result 34,14321 171,69127 0,69877 478,1795652883
Source 86,26095 187,58196 0,40258
Result 86,26095 187,58196 0,40258 265,9762582108
Source 4,70573 149,05859 0,43089
Result 4,70573 149,05859 0,43089 697,1195473051
Source 34,83820 167,47033 0,18253
Result 34,83820 167,47033 0,18253 270,0255145818
Source 56,19575 137,33325 0,70944
Result 56,19575 137,33325 0,70944 1590,3140383777
Source 25,78299 124,19618 0,03053
Result 25,78299 124,19618 0,03053 2282,8867721887
Source 81,25384 163,63499 0,01552
Result 81,25384 163,63499 0,01552 246,0035785670
Source 29,67182 126,28539 0,19766
Result 29,67182 126,28539 0,19766 254,4806412413
Source 18,72675 128,92234 0,72693
Result 18,72675 128,92234 0,72693 1340,2059362095
Source 11,60207 174,81655 0,55872
Result 11,60207 174,81655 0,55872 535,9777071349
Source 64,12893 187,94642 0,88856
Result 64,12893 187,94642 0,88856 308,2251376961
Source 33,42905 165,11616 0,78920
Result 33,42905 165,11616 0,78920 413,8341272801
Source 30,94024 107,62891 0,98436
Result 30,94024 107,62891 0,98436 159,0697885728
Source 69,53451 109,88573 0,70193
Result 69,53451 109,88573 0,70193 158,6233268572
Source 44,38043 176,12969 0,58964
Result 44,38043 176,12969 0,58964 335,7724524791
Source 30,26734 108,52633 0,74736
Result 30,26734 108,52633 0,74736 142,4340675780

Source 74,30112 159,86977 0,03283
Result 74,30112 159,86977 0,03283 212,9578195386
Source 12,19687 172,42635 0,93140
Result 12,19687 172,42635 0,93140 408,3586180485
Source 38,83579 130,77520 0,44537
Result 38,83579 130,77520 0,44537 240,7031759071
Source 19,31973 140,97797 0,65472
Result 19,31973 140,97797 0,65472 287,5480527921
Source 35,44842 111,00754 0,82885
Result 35,44842 111,00754 0,82885 185,2710994180
Source 24,36053 148,08803 0,22734
Result 24,36053 148,08803 0,22734 267,8580269156
Source 81,02152 107,44652 0,01813
Result 81,02152 107,44652 0,01813 73,0599689375
Source 64,30726 123,94890 0,06255
Result 64,30726 123,94890 0,06255 952,3804405127
Source 67,62404 133,19057 0,66944
Result 67,62404 133,19057 0,66944 459,2212843349
Source 89,32193 141,83499 0,80399
Result 89,32193 141,83499 0,80399 154,9190015709
Source 65,59982 138,34071 0,65244
Result 65,59982 138,34071 0,65244 152,5980718323
Source 9,16701 144,69054 0,88973
Result 9,16701 144,69054 0,88973 390,8018246422
Source 30,61729 197,33360 0,47487
Result 30,61729 197,33360 0,47487 542,4562393334
Source 6,28845 111,27427 0,30745
Result 6,28845 111,27427 0,30745 207,2275088429
Source 96,32088 125,08921 0,61908
Result 96,32088 125,08921 0,61908 79,7901175741
Source 9,69579 146,62655 0,54438
Result 9,69579 146,62655 0,54438 369,1243814283
Source 59,22494 119,23891 0,91903
Result 59,22494 119,23891 0,91903 153,3671058246
Source 48,77993 106,62467 0,98611
Result 48,77993 106,62467 0,98611 159,1894763147
Source 87,49071 150,69937 0,35681
Result 87,49071 150,69937 0,35681 157,9846806219
Source 31,20943 191,39195 0,26811
Result 31,20943 191,39195 0,26811 676,6746902592

Source 10,22692 149,85818 0,90755
Result 10,22692 149,85818 0,90755 294,7538873077
Source 14,04569 167,32617 0,33376
Result 14,04569 167,32617 0,33376 346,8212141021
Source 68,95475 160,63658 0,31172
Result 68,95475 160,63658 0,31172 606,7088303996
Source 62,21775 102,51299 0,20299
Result 62,21775 102,51299 0,20299 79,9963242324
Source 37,87768 159,69501 0,90347
Result 37,87768 159,69501 0,90347 928,7207410444
Source 13,49738 125,29010 0,70530
Result 13,49738 125,29010 0,70530 316,0618762695
Source 44,59694 163,04711 0,87431
Result 44,59694 163,04711 0,87431 591,1428091516
Source 94,11503 140,72752 0,68108
Result 94,11503 140,72752 0,68108 99,0347371062
Source 47,91310 167,34789 0,60062
Result 47,91310 167,34789 0,60062 264,8021197179
Source 54,15726 164,78987 0,40519
Result 54,15726 164,78987 0,40519 638,5472324842
Source 54,35452 198,12144 0,91420
Result 54,35452 198,12144 0,91420 1278,6591351149
Source 16,88627 160,94610 0,23669
Result 16,88627 160,94610 0,23669 2090,8326102041
Source 80,07315 162,19407 0,44464
Result 80,07315 162,19407 0,44464 266,6419414525

Simple threads
Source 15,71364 164,61244 0,45334
Result 15,71364 164,61244 0,45334 318,2252240896
Source 98,79890 118,05611 0,05519
Result 98,79890 118,05611 0,05519 97,2633589458
Source 95,58991 109,03784 0,24418
Result 95,58991 109,03784 0,24418 41,9158718038
Source 10,59633 155,14045 0,87301
Result 10,59633 155,14045 0,87301 292,0179522677
Source 91,97368 189,17019 0,47249
Result 91,97368 189,17019 0,47249 2268,9243420048
Source 87,53414 146,74586 0,19105
Result 87,53414 146,74586 0,19105 569,2371275548

Source 39,80704 121,13975 0,99163
Result 39,80704 121,13975 0,99163 223,2240136893
Source 26,93903 181,14960 0,62742
Result 26,93903 181,14960 0,62742 303,9490498281
Source 60,22822 118,19123 0,34865
Result 60,22822 118,19123 0,34865 140,5578109626
Source 56,21510 187,33665 0,25441
Result 56,21510 187,33665 0,25441 319,2440922366
Source 11,50709 133,68238 0,69699
Result 11,50709 133,68238 0,69699 399,5870306288
Source 21,89275 150,21816 0,17014
Result 21,89275 150,21816 0,17014 298,3154035119
Source 3,80502 196,27619 0,66877
Result 3,80502 196,27619 0,66877 493,9478744904
Source 95,82232 116,67627 0,59426
Result 95,82232 116,67627 0,59426 89,4323347324
Source 40,64993 140,84234 0,89786
Result 40,64993 140,84234 0,89786 312,8530398769
Source 73,90308 108,45109 0,30391
Result 73,90308 108,45109 0,30391 536,8596900187
Source 75,01701 163,58348 0,13545
Result 75,01701 163,58348 0,13545 583,9164711374
Source 67,73834 182,88519 0,47661
Result 67,73834 182,88519 0,47661 307,2622290583
Source 92,90576 154,90199 0,14931
Result 92,90576 154,90199 0,14931 201,7747122557
Source 14,60715 107,54153 0,25630
Result 14,60715 107,54153 0,25630 179,1293791615
Source 62,30405 188,52284 0,84817
Result 62,30405 188,52284 0,84817 348,7222391478
Source 56,14330 139,71020 0,60893
Result 56,14330 139,71020 0,60893 187,1850968719
Source 10,42969 170,38819 0,32276
Result 10,42969 170,38819 0,32276 1671,4841037025
Source 61,11217 126,81975 0,28423
Result 61,11217 126,81975 0,28423 139,3055243616
Source 17,14028 175,83937 0,74472
Result 17,14028 175,83937 0,74472 416,0433603724
Source 92,67620 168,76868 0,94972
Result 92,67620 168,76868 0,94972 232,7517466682

Source 37,40187 128,60175 0,31497
Result 37,40187 128,60175 0,31497 207,1177370043
Source 45,53942 108,19270 0,81368
Result 45,53942 108,19270 0,81368 212,0202808810
Source 90,70709 100,08832 0,70282
Result 90,70709 100,08832 0,70282 20,2504955587
Source 3,36180 139,68346 0,30331
Result 3,36180 139,68346 0,30331 275,6250645087
Source 80,33332 132,53743 0,66724
Result 80,33332 132,53743 0,66724 106,4064891202
Source 91,83398 153,97425 0,61449
Result 91,83398 153,97425 0,61449 200,5331071931
Source 30,33169 114,50041 0,80184
Result 30,33169 114,50041 0,80184 401,0038268341
Source 27,80307 158,10839 0,19972
Result 27,80307 158,10839 0,19972 323,7390119794
Source 53,13728 162,19466 0,59470
Result 53,13728 162,19466 0,59470 317,0126036388
Source 89,84004 137,60598 0,51035
Result 89,84004 137,60598 0,51035 117,3814419696
Source 85,69139 185,23785 0,24474
Result 85,69139 185,23785 0,24474 391,4427053069
Source 60,63941 151,64442 0,55979
Result 60,63941 151,64442 0,55979 201,6575081948
Source 56,96631 140,27882 0,92582
Result 56,96631 140,27882 0,92582 266,0885298482
Source 16,64185 177,26864 0,12397
Result 16,64185 177,26864 0,12397 365,7327626237
Source 74,78339 101,04089 0,44010
Result 74,78339 101,04089 0,44010 57,5670225447
Result 65,27031 143,64637 0,64538 177,5280008100
Source 65,27031 143,64637 0,64538
Source 44,81490 198,29312 0,45204
Result 44,81490 198,29312 0,45204 488,6626888956
Source 55,17027 122,38425 0,15736
Result 55,17027 122,38425 0,15736 141,0811886644
Source 59,66030 108,26512 0,74437
Result 59,66030 108,26512 0,74437 947,5357760658
Source 9,84480 164,92036 0,79472
Result 9,84480 164,92036 0,79472 532,5238963570

Source 20,88777 193,02119 0,18757
Result 20,88777 193,02119 0,18757 634,9764648116
Source 60,07498 103,00877 0,70734
Result 60,07498 103,00877 0,70734 95,3838650309
Source 81,93122 121,30926 0,36420
Result 81,93122 121,30926 0,36420 202,8403602053
Source 75,51640 153,31178 0,44301
Result 75,51640 153,31178 0,44301 199,9500001531
Source 24,73381 123,40025 0,03936
Result 24,73381 123,40025 0,03936 222,1672547395
Source 20,86228 192,11543 0,11607
Result 20,86228 192,11543 0,11607 477,5609073144
Source 8,02843 158,43368 0,91901
Result 8,02843 158,43368 0,91901 332,5809665248
Source 86,96381 191,18394 0,92805
Result 86,96381 191,18394 0,92805 346,4335351403
Source 50,14560 197,78485 0,38716
Result 50,14560 197,78485 0,38716 5476,0894687508
Source 72,29854 130,76869 0,85691
Result 72,29854 130,76869 0,85691 123,3308009178
Source 34,23980 152,96950 0,16366
Result 34,23980 152,96950 0,16366 5825,5151611488
Source 15,17711 130,78735 0,58115
Result 15,17711 130,78735 0,58115 228,2698938940
Source 48,82452 101,60136 0,07382
Result 48,82452 101,60136 0,07382 211,1662050919
Source 2,07928 166,18882 0,95530
Result 2,07928 166,18882 0,95530 462,8906729360
Source 1,50675 189,91834 0,37131
Result 1,50675 189,91834 0,37131 420,8579165415
Source 2,50966 157,51958 0,60310
Result 2,50966 157,51958 0,60310 312,4688894630
Source 68,81163 115,39295 0,30573
Result 68,81163 115,39295 0,30573 110,2084420304
Source 5,76385 152,70708 0,44486
Result 5,76385 152,70708 0,44486 319,4072286466
Source 69,98779 117,94261 0,87638
Result 69,98779 117,94261 0,87638 206,0252846607
Source 59,99562 141,07579 0,22534
Result 59,99562 141,07579 0,22534 165,6837433695

Source 77,68400 133,25533 0,96336
Result 77,68400 133,25533 0,96336 116,8290835899
Source 38,97574 165,61155 0,35861
Result 38,97574 165,61155 0,35861 427,4079564008
Source 3,38782 186,74659 0,27775
Result 3,38782 186,74659 0,27775 590,9250227279
Source 86,93307 124,48985 0,65628
Result 86,93307 124,48985 0,65628 132,3532367997
Source 20,51680 131,66295 0,82199
Result 20,51680 131,66295 0,82199 227,9508413292
Source 16,86855 177,88361 0,75542
Result 16,86855 177,88361 0,75542 584,2765979019
Source 13,05294 159,84795 0,80204
Result 13,05294 159,84795 0,80204 289,6645850352
Source 62,73274 101,06614 0,31279
Result 62,73274 101,06614 0,31279 137,0689529830
Source 62,61092 138,68602 0,18264
Result 62,61092 138,68602 0,18264 152,8763814417
Source 10,85751 149,82496 0,28150
Result 10,85751 149,82496 0,28150 267,4329082343
Source 11,83091 152,74003 0,32711
Result 11,83091 152,74003 0,32711 313,3982495983
Source 61,03244 129,20190 0,83681
Result 61,03244 129,20190 0,83681 270,6069788951
Source 66,98630 153,59798 0,41309
Result 66,98630 153,59798 0,41309 208,7380973527
Source 92,61672 181,57538 0,76716
Result 92,61672 181,57538 0,76716 190,3955548392
Source 27,69730 146,44466 0,56564
Result 27,69730 146,44466 0,56564 322,4258708656
Source 36,07701 135,10026 0,39716
Result 36,07701 135,10026 0,39716 189,9830415858
Source 56,64079 166,87369 0,89907
Result 56,64079 166,87369 0,89907 294,8907333168
Source 75,34195 178,52101 0,35865
Result 75,34195 178,52101 0,35865 238,7162650207
Source 27,14716 126,45570 0,90895
Result 27,14716 126,45570 0,90895 224,4848465350
Source 31,36400 153,37386 0,77646
Result 31,36400 153,37386 0,77646 403,4687269453

Source 11,57473 130,57825 0,90566
Result 11,57473 130,57825 0,90566 264,2318454744
Source 2,27908 179,59333 0,68518
Result 2,27908 179,59333 0,68518 524,0472507167
Source 98,35239 197,07308 0,43092
Result 98,35239 197,07308 0,43092 741,8093860239
Source 60,55487 155,53914 0,74326
Result 60,55487 155,53914 0,74326 243,4096497491
Source 50,58109 129,14638 0,84875
Result 50,58109 129,14638 0,84875 161,8898020702
Source 32,51815 198,45871 0,56112
Result 32,51815 198,45871 0,56112 586,7014145179
Source 44,08473 121,12758 0,55209
Result 44,08473 121,12758 0,55209 163,7727241968
Source 10,65195 116,40122 0,20927
Result 10,65195 116,40122 0,20927 269,0675955978
Source 47,80125 100,55249 0,47676
Result 47,80125 100,55249 0,47676 157,3528085167
Source 74,05310 127,59862 0,72332
Result 74,05310 127,59862 0,72332 114,7401922430
Source 34,40432 135,09615 0,86389
Result 34,40432 135,09615 0,86389 302,1899312457
Source 17,44085 168,32939 0,67179
Result 17,44085 168,32939 0,67179 385,6393957697
Source 34,04645 157,87132 0,46244
Result 34,04645 157,87132 0,46244 520,2217188788
Source 14,10704 155,28048 0,10361
Result 14,10704 155,28048 0,10361 1537,5194145284

Complicated threads
Source 59,3429 174,9632 0,2895
Result 59,3429 174,9632 0,2895 290,75457710
Source 34,0347 164,0294 0,3167
Result 34,0347 164,0294 0,3167 629,75319135
Source 0,9745 138,2626 0,0482
Result 0,9745 138,2626 0,0482 278,44275142
Source 53,0502 107,3910 0,4849
Result 53,0502 107,3910 0,4849 166,18051389
Source 69,7581 198,1566 0,5785
Result 69,7581 198,1566 0,5785 447,67157076

Source 30,4089 166,0599 0,9205
Result 30,4089 166,0599 0,9205 328,50394562
Source 47,7147 125,1959 0,5546
Result 47,7147 125,1959 0,5546 585,68009799
Source 85,0927 178,9716 0,2954
Result 85,0927 178,9716 0,2954 301,70154225
Source 59,3944 114,2827 0,2125
Result 59,3944 114,2827 0,2125 114,32711555
Source 72,2791 129,6344 0,7430
Result 72,2791 129,6344 0,7430 349,02114977
Source 58,0749 103,7285 0,2341
Result 58,0749 103,7285 0,2341 1319,27880423
Source 83,7266 175,1997 0,5840
Result 83,7266 175,1997 0,5840 201,63341975
Source 87,3078 196,4657 0,0499
Result 87,3078 196,4657 0,0499 419,44199435
Source 95,5795 145,0702 0,2223
Result 95,5795 145,0702 0,2223 126,39865415
Source 18,2039 124,4303 0,6260
Result 18,2039 124,4303 0,6260 305,75841189
Source 76,4809 162,5601 0,4027
Result 76,4809 162,5601 0,4027 234,15348155
Source 87,8469 122,1297 0,0531
Result 87,8469 122,1297 0,0531 113,00994473
Source 93,8857 168,1248 0,2727
Result 93,8857 168,1248 0,2727 375,33578430
Source 89,2273 100,4125 0,5031
Result 89,2273 100,4125 0,5031 23,81859570
Source 81,4629 147,6741 0,9375
Result 81,4629 147,6741 0,9375 194,76309457
Source 88,0871 136,4546 0,3694
Result 88,0871 136,4546 0,3694 177,42753473
Source 55,4792 194,4859 0,4529
Result 55,4792 194,4859 0,4529 505,47155818
Source 67,0646 120,9786 0,6583

Process finished with exit code 0