Name: Artur

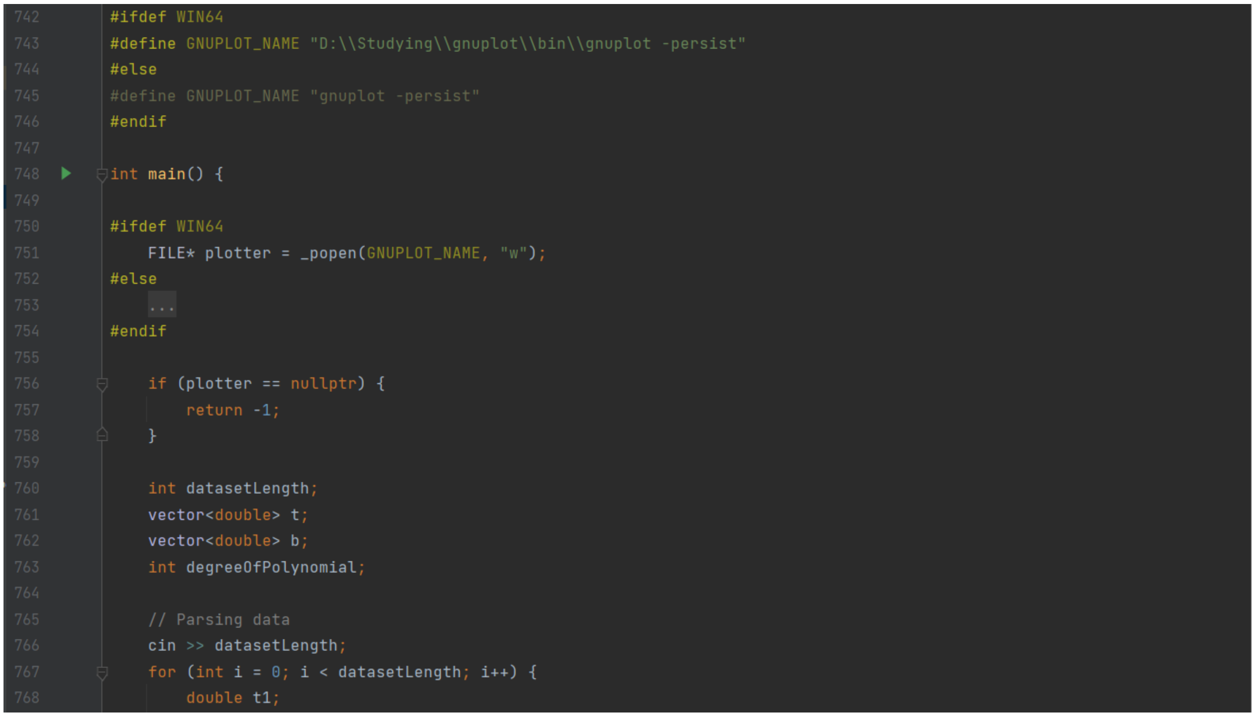
Surname: Mukhutdinov

Group number: CS-03

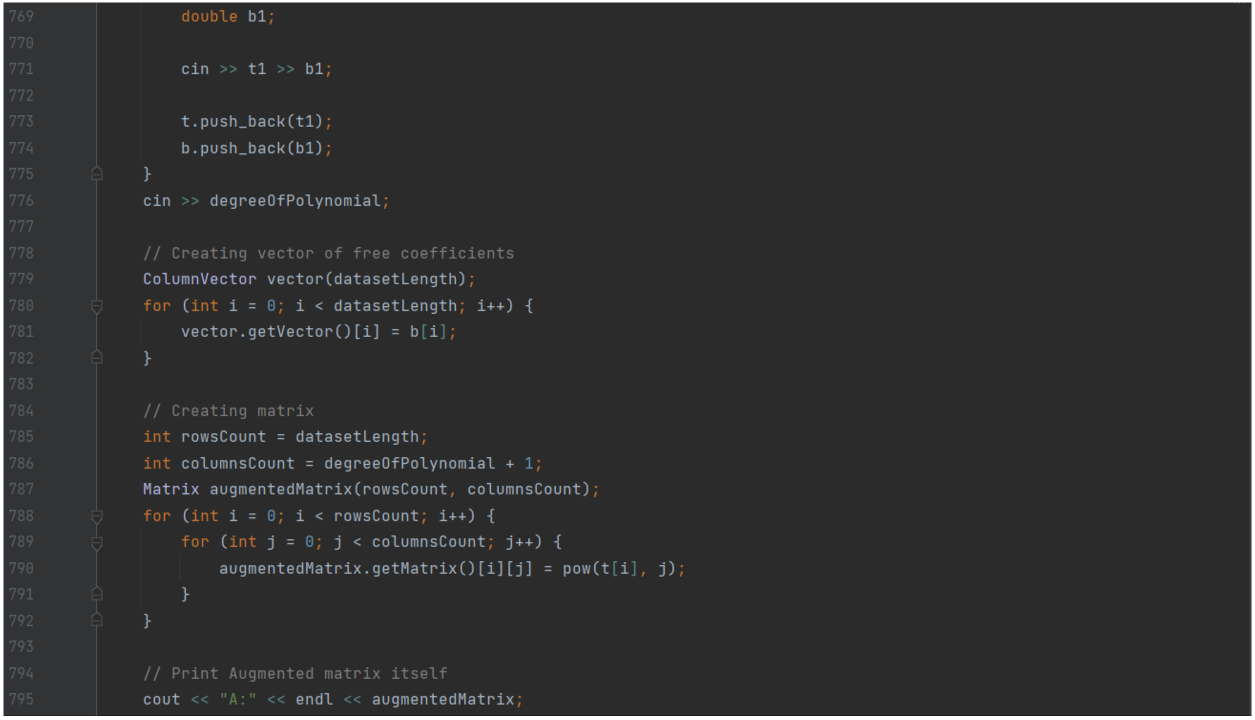
e-mail: a.mukhutdinov@innopolis.university

cout << shortRepontOnLSA << endl;

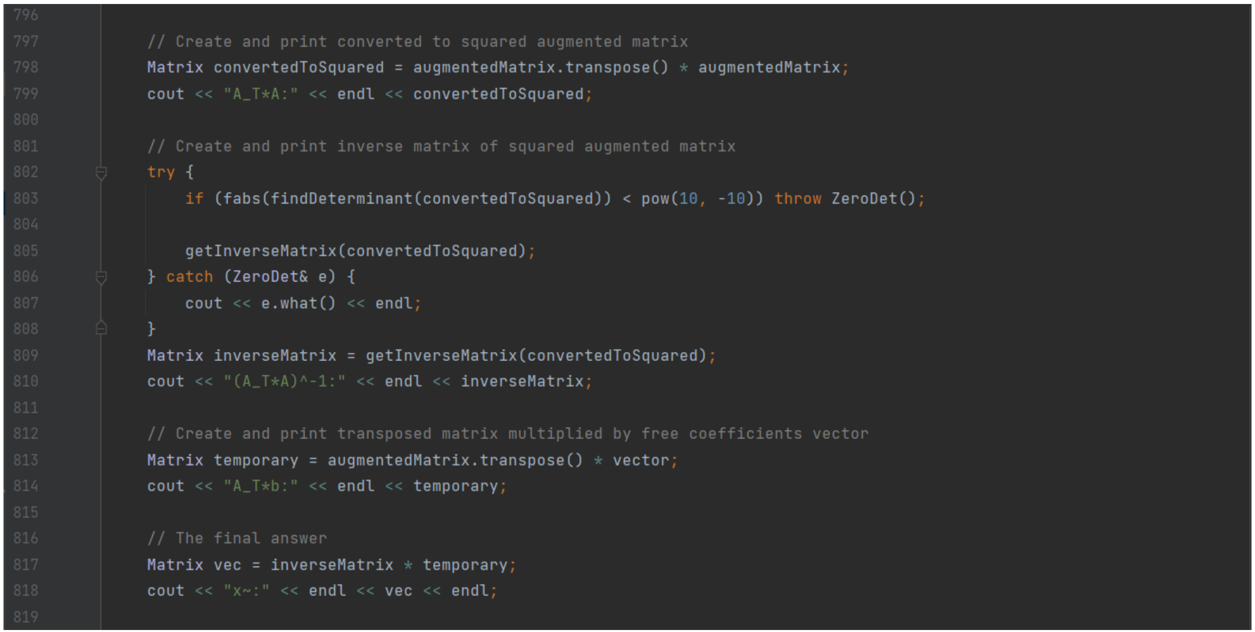
The main implementation of the LSA algorithm By the way, the names of the function and variables describe themselves, so for now I attach only the int main() function from my implementation. (There are can be some changes in code during the plotting procedure to, for instance, make LEGEND visible if graphs are intersect with it)



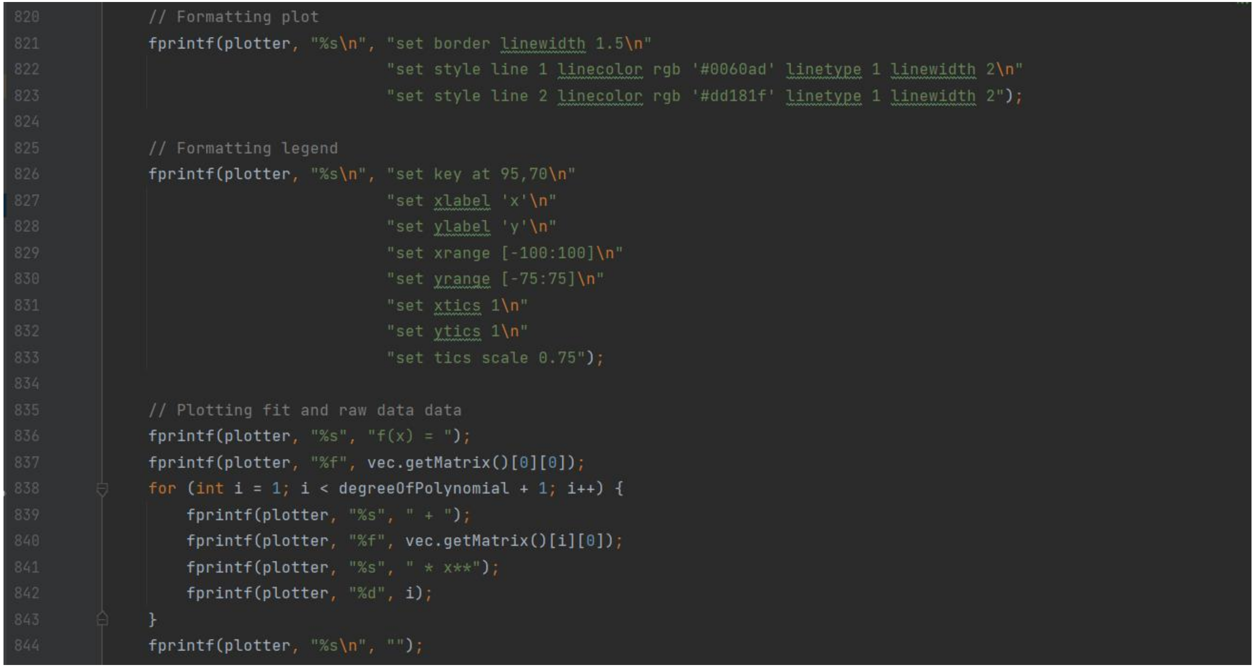
```cpp
#ifdef WIN64
#define GNUPLOT_NAME "D:\\Studying\\gnuplot\\bin\\gnuplot -persist"
#else
#define GNUPLOT_NAME "gnuplot -persist"
#endif

int main() {

#ifdef WIN64
    FILE* plotter = _popen(GNUPLOT_NAME, "w");
#else
    ...
#endif

    if (plotter == nullptr) {
        return -1;
    }

    int datasetLength;
    vector<double> t;
    vector<double> b;
    int degreeOfPolynomial;

    // Parsing data
    cin >> datasetLength;
    for (int i = 0; i < datasetLength; i++) {
        double t1;
```



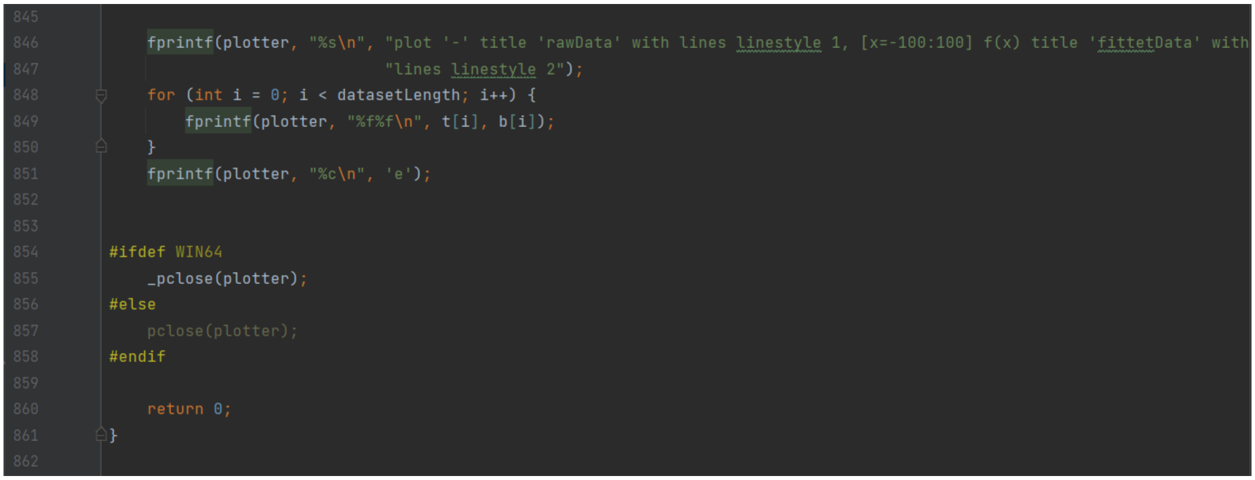
```cpp
        double b1;

        cin >> t1 >> b1;

        t.push_back(t1);
        b.push_back(b1);
    }
    cin >> degreeOfPolynomial;

    // Creating vector of free coefficients
    ColumnVector vector(datasetLength);
    for (int i = 0; i < datasetLength; i++) {
        vector.getVector()[i] = b[i];
    }

    // Creating matrix
    int rowsCount = datasetLength;
    int columnsCount = degreeOfPolynomial + 1;
    Matrix augmentedMatrix(rowsCount, columnsCount);
    for (int i = 0; i < rowsCount; i++) {
        for (int j = 0; j < columnsCount; j++) {
            augmentedMatrix.getMatrix()[i][j] = pow(t[i], j);
        }
    }

    // Print Augmented matrix itself
    cout << "A:" << endl << augmentedMatrix;
```



```cpp

    // Create and print converted to squared augmented matrix
    Matrix convertedToSquared = augmentedMatrix.transpose() * augmentedMatrix;
    cout << "A_T*A:" << endl << convertedToSquared;

    // Create and print inverse matrix of squared augmented matrix
    try {
        if (fabs(findDeterminant(convertedToSquared)) < pow(10, -10)) throw ZeroDet();

        getInverseMatrix(convertedToSquared);
    } catch (ZeroDet& e) {
        cout << e.what() << endl;
    }
    Matrix inverseMatrix = getInverseMatrix(convertedToSquared);
    cout << "(A_T*A)^-1:" << endl << inverseMatrix;

    // Create and print transposed matrix multiplied by free coefficients vector
    Matrix temporary = augmentedMatrix.transpose() * vector;
    cout << "A_T*b:" << endl << temporary;

    // The final answer
    Matrix vec = inverseMatrix * temporary;
    cout << "x~:" << endl << vec << endl;

```



```cpp
    // Formatting plot
    fprintf(plotter, "%s\n", "set border linewidth 1.5\n"
                             "set style line 1 linecolor rgb '#0060ad' linetype 1 linewidth 2\n"
                             "set style line 2 linecolor rgb '#dd181f' linetype 1 linewidth 2");

    // Formatting legend
    fprintf(plotter, "%s\n", "set key at 95,70\n"
                             "set xlabel 'x'\n"
                             "set ylabel 'y'\n"
                             "set xrange [-100:100]\n"
                             "set yrange [-75:75]\n"
                             "set xtics 1\n"
                             "set ytics 1\n"
                             "set tics scale 0.75");

    // Plotting fit and raw data data
    fprintf(plotter, "%s", "f(x) = ");
    fprintf(plotter, "%f", vec.getMatrix()[0][0]);
    for (int i = 1; i < degreeOfPolynomial + 1; i++) {
        fprintf(plotter, "%s", " + ");
        fprintf(plotter, "%f", vec.getMatrix()[i][0]);
        fprintf(plotter, "%s", " * x**");
        fprintf(plotter, "%d", i);
    }
    fprintf(plotter, "%s\n", "");
```



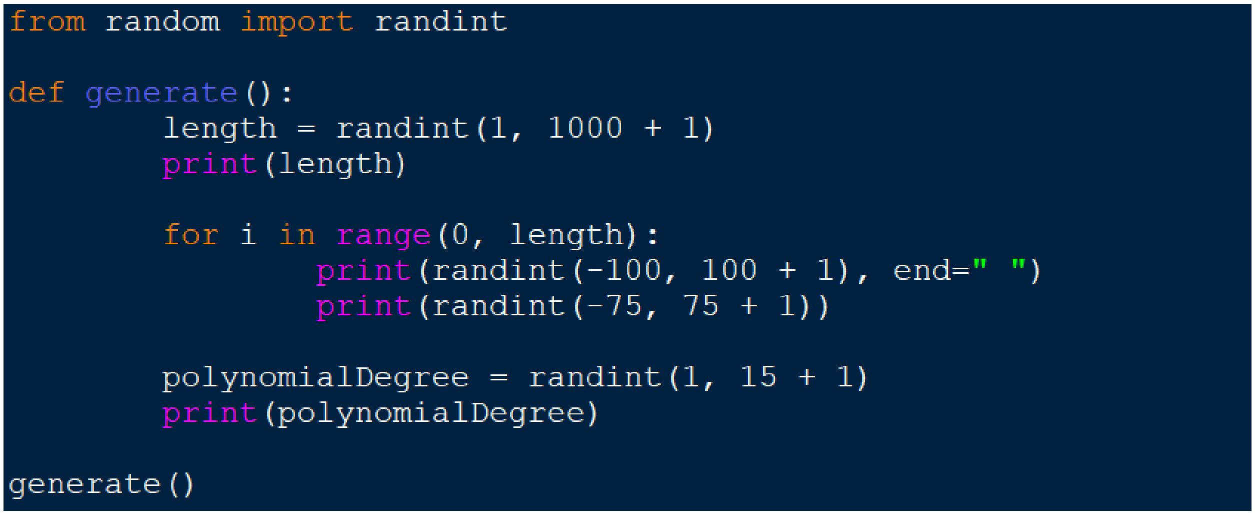
```
    fprintf(plotter, "%s\n", "plot '-' title 'rawData' with lines linestyle 1, [x=-100:100] f(x) title 'fittetData' with
                             "lines linestyle 2");
    for (int i = 0; i < datasetLength; i++) {
        fprintf(plotter, "%f%f\n", t[i], b[i]);
    }
    fprintf(plotter, "%c\n", 'e');


#ifdef WIN64
    _pclose(plotter);
#else
    pclose(plotter);
#endif

    return 0;
}
```

A set of points will be generated through the Python programming language. I created a pythonTest.py program that generates a fixed amount of number N, which is input by the user. There are described two sets of points: the first one contains only the integer points, and the second one contains all values ∈ R.

The chosen range for both tests: x[-100, 100], y[-75, 75]; polynomial degree[1:15]; length of the input[1:1000]

cout << Set of Points I << endl;

Python code for generator



```
from random import randint

def generate():
        length = randint(1, 1000 + 1)
        print(length)

        for i in range(0, length):
                print(randint(-100, 100 + 1), end=" ")
                print(randint(-75, 75 + 1))

        polynomialDegree = randint(1, 15 + 1)
        print(polynomialDegree)

generate()
```

Obtained input from generator (according to the input format from the Yandex.Contest, assignment 2, Task 1):

86

81 -58

13 19

23 62

88 4

-35 58

86 -36

41 -67

-14 14

99 -6

-20 71

13 69

20 -45

-19 72

86 72

-4 -11

-52 33

34 -71

71 -53

63 57

-40 47

99 -56

-43 -65

-4 -21

71 -24

41 47

32 22

73 45

27 5

3 -39

51 12

-67 21

-39 45

30 -35

-38 -42

-61 11

69 -19

-25 72

10 -18

91 22

-81 -58

-95 1

-79 64

-61 13

53 47

-67 62

75 57

-68 -58

60 1

55 72

7 22

92 62

-63 62

45 41

90 -41

76 -43

62 -36

45 -35

-69 26

83 -16

-38 -9

67 -54

-15 -58

-48 -60

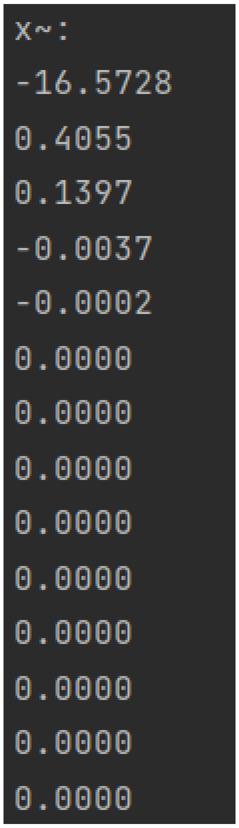
-61 4

94 61

37 -30

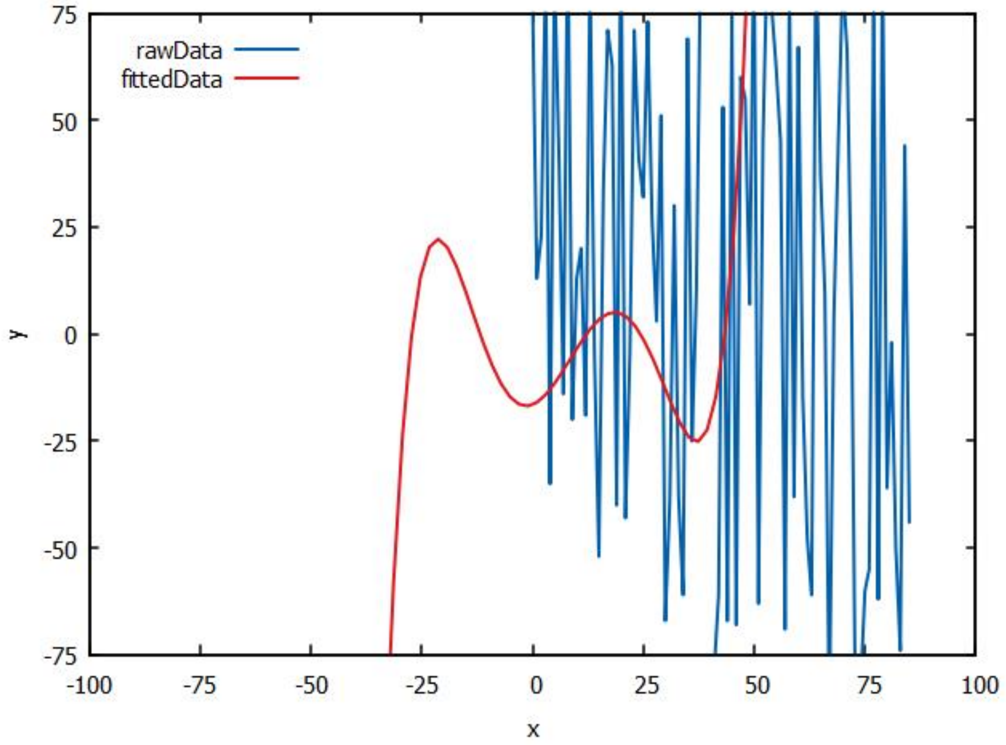
```
9 36

-91 -68

4 -33

44 49

83 -26

66 18

5 -34

-98 -39

-84 -28

-60 24

-55 -7

77 12

-62 64

92 1

-36 51

-2 -73

-50 -35

-74 -17

44 -70

-44 15

13
```

Obtained output of the c++ program (only the last vector):



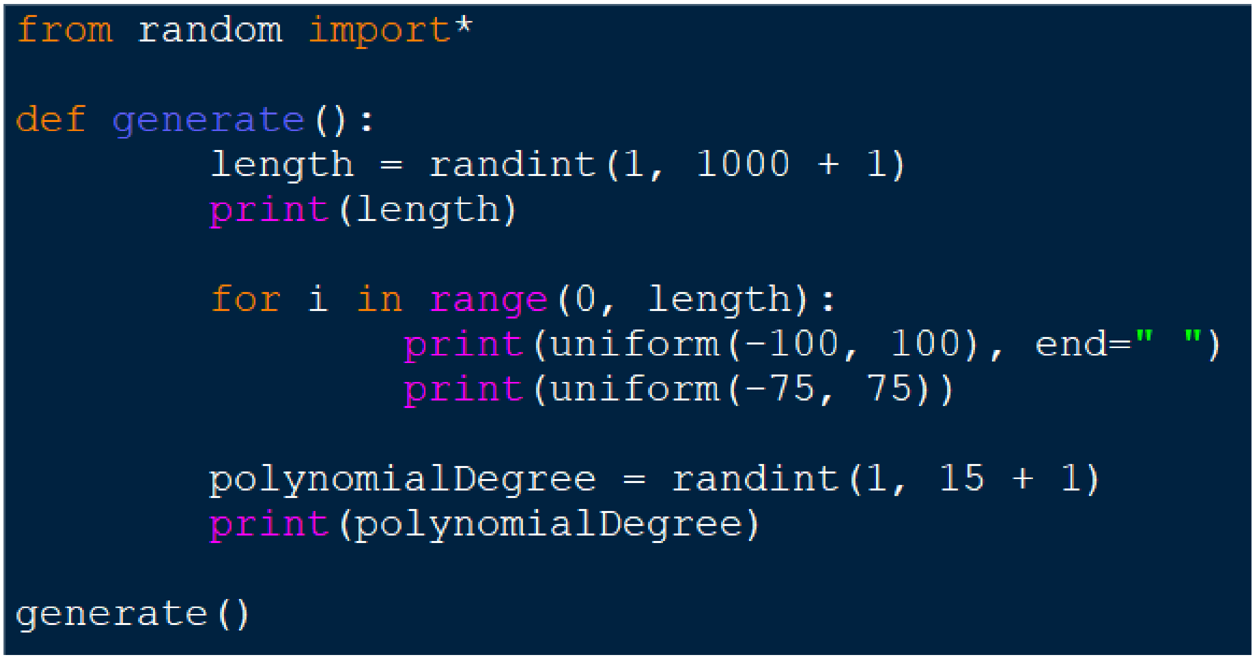
```
x~:
-16.5728
0.4055
0.1397
-0.0037
-0.0002
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
```

The graph from GNUplot:



cout << Set of Points II << endl;

Python code for generator



```python
from random import*

def generate():
        length = randint(1, 1000 + 1)
        print(length)

        for i in range(0, length):
                print(uniform(-100, 100), end=" ")
                print(uniform(-75, 75))

        polynomialDegree = randint(1, 15 + 1)
        print(polynomialDegree)

generate()
```

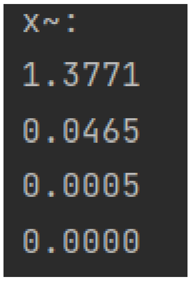
Obtained input from generator (according to the input format from the Yandex.Contest, assignment 2, Task 1):

```
50
96.18434080093161 62.54788847330627
39.24331293282583 40.07087775375325
18.940314629503447 44.288097991180905
78.4455594283952 63.82623173170276
0.16578642005642052 72.63132783468495
41.31211461819743 -2.71451676708287
-54.00471115861032 -8.352723921778363
10.739117971461226 57.88709735896546
13.174770003810309 -44.28159134224028
-86.56823032507172 -18.757530954397808
```

-98.99330877037646	29.864294002797863
77.60120831976101	-29.469692976292542
-92.16232138161757	57.70951768124948
-7.144582066051484	-3.4466414087787456
-25.5634583767324	37.525568169767155
-52.49866916085033	51.54681489556724
-22.498015010038557	-24.23610697060719
27.784108336271032	40.3735051917306
-10.491893793905405	-72.21661995525105
-26.011731588621274	19.731654165236833
52.9597843052278	9.785176140977228
-79.39794493247068	-41.29823712551018
-16.127021975889804	-27.468437143651535
17.235517441549433	-34.876171359588085
8.71886618466364	14.494787017545079
77.69871494403637	-38.274456636230106
-12.788502153900865	-69.02118996545276
-55.024438127375255	40.5393765442944
62.9137867926182	43.61456066627419
-94.51712338835108	-74.19975518574246
46.908805231706054	-9.289501348748374
-73.62057471011295	48.97858596903821
-53.16816626747256	-23.937849336015226
-47.820380036412956	44.67440661368245
-69.65029428106484	-63.73840459729146
-14.637327892614422	26.783208130943436

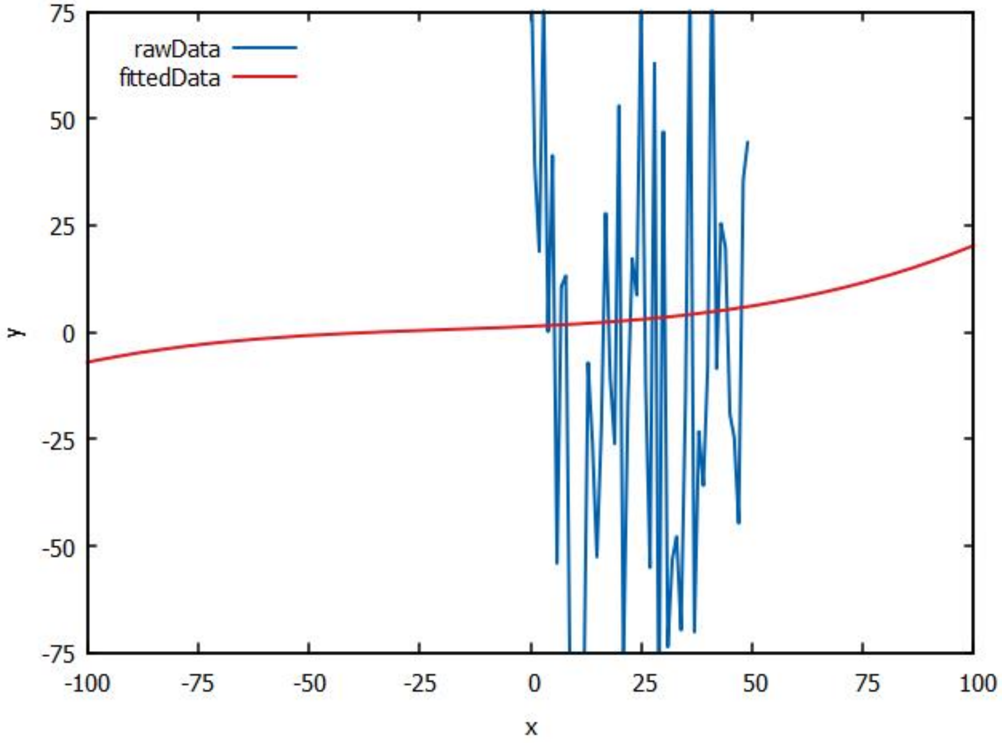
```
82.47098205211745 0.461451378266986

-70.08935955549674 -43.47923506697511

-23.264308708835316 3.314819718166504

-35.85708918654653 19.786402997419955

-8.371232296424893 -42.13292616532124

92.5875221693922 -9.107338441735152

-8.467386984756573 -59.960779571447915

25.423609521968544 36.20717003190754

19.91657162385347 -31.746351388980607

-19.14464674351406 8.945250239921933

-24.574795754117005 70.4162139390674

-44.650995095215485 -36.012307630366145

35.574412477040255 -6.180275365429864

44.47011150513481 -13.830120892994707

3
```

Obtained output of the c++ program (only the last vector):



```
x~:
1.3771
0.0465
0.0005
0.0000
```

The graph from GNUplot:


rawData
fittedData
y
x
75
50
25
0
-25
-50
-75
-100
-75
-50
-25
0
25
50
75
100