#include <iostream>

#include <vector>

#include <fstream>

#include <string>

using namespace std;

typedef vector<double> functionType(double, vector<double>);

int n;

// y" + p(t)\*y' + q(x)\*y = f(t)

double f(double t) {

if (n == 1) {

return 0;

}

else if (n == 2) {

return 6 \* t/(pow(t, 2) - 1);

}

}

double p(double t) {

if (n == 1) {

return 4 \* t / (2 \* t + 1);

}

else if (n == 2) {

return 4 \* t / (pow(t, 2) - 1);

}

}

double q(double t) {

if (n == 1) {

return -4 / (2 \* t + 1);

}

else if (n == 2) {

return -2 / (pow(t, 2) - 1);

}

}

// y'(a) - al\_1\*y(a) = al\_2

// y'(b) - bt\_1\*y(b) = bt\_2

double a; double al\_1; double al\_2;

double b; double bt\_1; double bt\_2;

void setBorders() {

if (n == 1) {

a = 0; al\_1 = 1; al\_2 = -2;

b = 2; bt\_1 = 1; bt\_2 = -1.073262555554937;

}

else if (n == 2) {

a = 2; al\_1 = 1; al\_2 = -1.234721044665223;

b = 7; bt\_1 = 1; bt\_2 = -2.420465655758776;

}

}

vector<double> sysU(double t, vector<double> v) {

vector<double> tmp;

tmp.resize(2);

tmp[0] = v[1];

tmp[1] = f(t) - q(t)\*v[0] -p(t)\*v[1];

return tmp;

}

vector<double> initCon(double xi) {

vector<double> tmp;

tmp.resize(2);

tmp[0] = xi;

tmp[1] = al\_2 + al\_1 \* xi;

return tmp;

}

// сложение 2-х векторов

vector<double> addVectors(vector<double> a, vector<double> b) {

for (int i = 0; i < a.size(); ++i) {

a[i] += b[i];

}

return a;

}

// сложение 4-х векторов

vector<double> add4Vectors(vector<double> a, vector<double> b, vector<double> c, vector<double> d) {

for (int i = 0; i < a.size(); ++i) {

a[i] += b[i] + c[i] + d[i];

}

return a;

}

// умножение ветора на число

vector<double> multVectorByNumber(vector<double> a, double b) {

for (int i = 0; i < a.size(); ++i) {

a[i] \*= b;

}

return a;

}

//k\_1 = f(t\_n, y\_n)

vector<double> coefK1(double t, vector<double> v) {

return sysU(t, v);

}

//k\_2 = f(t + h, y\_n + h\*k\_1)

vector<double> coefK2(double t, vector<double> v, vector<double> k, double h) {

return sysU(t + h/2, addVectors(v, multVectorByNumber(k, h/2)));

}

//k\_2 = f(t + h, y\_n + h\*k\_1)

vector<double> coefK4(double t, vector<double> v, vector<double> k, double h) {

return sysU(t + h, addVectors(v, multVectorByNumber(k, h)));

}

// метод рунге 1-го прядка

vector<double> runge(double t, vector<double> v, double h) {

vector<double> k1 = coefK1(t, v);

vector<double> k2 = coefK2(t, v, k1, h);

vector<double> k3 = coefK2(t, v, k2, h);

vector<double> k4 = coefK4(t, v, k3, h);

return addVectors(v, multVectorByNumber(add4Vectors(k1, multVectorByNumber(k2, 2), multVectorByNumber(k3, 2), k4), h / 6));

}

vector<vector<double>> adamas(double xi) {

vector<vector<double>> answer;

answer.resize(3);

double h = 0.01;

//нулевой шаг

double t = a;

vector<double> v0 = initCon(xi);

answer[0].push\_back(t);

answer[1].push\_back(v0[0]);

answer[2].push\_back(v0[1]);

//первый шаг

t += h;

vector<double> v1 = runge(t, v0, h);

answer[0].push\_back(t);

answer[1].push\_back(v1[0]);

answer[2].push\_back(v1[1]);

//второй шаг

t += h;

vector<double> v2 = runge(t, v1, h);

answer[0].push\_back(t);

answer[1].push\_back(v2[0]);

answer[2].push\_back(v2[1]);

//третий шаг

t += h;

vector<double> v3 = runge(t, v2, h);

answer[0].push\_back(t);

answer[1].push\_back(v3[0]);

answer[2].push\_back(v3[1]);

//Метод Адамса

while (t <= b) {

t += h;

vector<double> vNext = addVectors(v3, multVectorByNumber(add4Vectors(multVectorByNumber(sysU(t, v3), 55),

multVectorByNumber(sysU(t, v2), -59), multVectorByNumber(sysU(t, v1), 37), multVectorByNumber(sysU(t, v0), -9)), h / 24));

answer[0].push\_back(t);

answer[1].push\_back(vNext[0]);

answer[2].push\_back(vNext[1]);

v0 = v1;

v1 = v2;

v2 = v3;

v3 = vNext;

}

return answer;

}

double test1;

double test2;

double test3;

double test4;

pair<double, vector<vector<double>>> fi(double x) {

vector<vector<double>> tmp = adamas(x);

//vector<vector<double>> tmp2;

//tmp2.push\_back(tmp[0]);

//tmp2.push\_back(tmp[1]);

test1 = tmp[1][tmp[1].size() - 1];

test2 = tmp[2][tmp[2].size() - 1];

test3 = tmp[1][tmp[1].size() - 1] - bt\_1 \* tmp[2][tmp[2].size() - 1] - bt\_2;

return make\_pair( (tmp[2][tmp[2].size() - 1] - bt\_1 \* tmp[1][tmp[1].size() - 1] - bt\_2), /\*tmp2\*/ tmp );

}

vector<vector<double>> solve() {

pair<double, vector<vector<double>>> result;

double eps = 0.1;

double xi\_a = -100;

double xi\_b = 100;

result = fi(xi\_a);

/\*

pair<double, vector<vector<double>>> closestToZero = result;

double h = xi\_a + eps;

while (h <= xi\_b) {

result = fi(h);

if (abs(closestToZero.first) > abs(result.first)) {

closestToZero = result;

}

cout << h << " " << result.first << endl;

h += eps;

}

cout << "closest " << closestToZero.first << endl;

return closestToZero.second;\*/

double l = xi\_b - xi\_a;

result = fi(xi\_a);

double f\_a = result.first;

//if (abs(f\_a) < eps) {

// cout << f\_a << endl;

// return result.second;

//}

result = fi(xi\_b);

double f\_b = result.first;

//if (abs(f\_b) < eps) {

// cout << f\_b << endl;

// return result.second;

//}

while (l > eps) {

double xi\_c = (xi\_a + xi\_b) / 2;

result = fi(xi\_c);

double f\_c = result.first;

//if (abs(f\_c) < eps) {

// cout << f\_c << endl;

// return result.second;

//}

if (f\_a \* f\_c < 0) {

xi\_b = xi\_c;

f\_b = f\_c;

//cout << "-" << endl;

}

else if (f\_c \* f\_b < 0) {

xi\_a = xi\_c;

f\_a = f\_c;

//cout << "-" << endl;

}

else {

cout << "f\_a " << f\_a << endl;

cout << "f\_c " << f\_c << endl;

cout << "f\_b " << f\_b << endl;

}

l = abs(xi\_b - xi\_a);

cout << "test " << l << endl;

}

cout << f\_a << endl;

return result.second;

}

int main() {

cout << "Enter file name: ";

cin >> n;

setBorders();

ofstream output;

output.open("answer.txt");

vector<vector<double>> answer = solve();

for (int i = 0; i < answer[0].size(); ++i) {

output << answer[0][i] << ' ' << answer[1][i] << ' ' << answer[2][i] << endl;

}

output.close();

system("pause");

}