* 1. **Introduction**

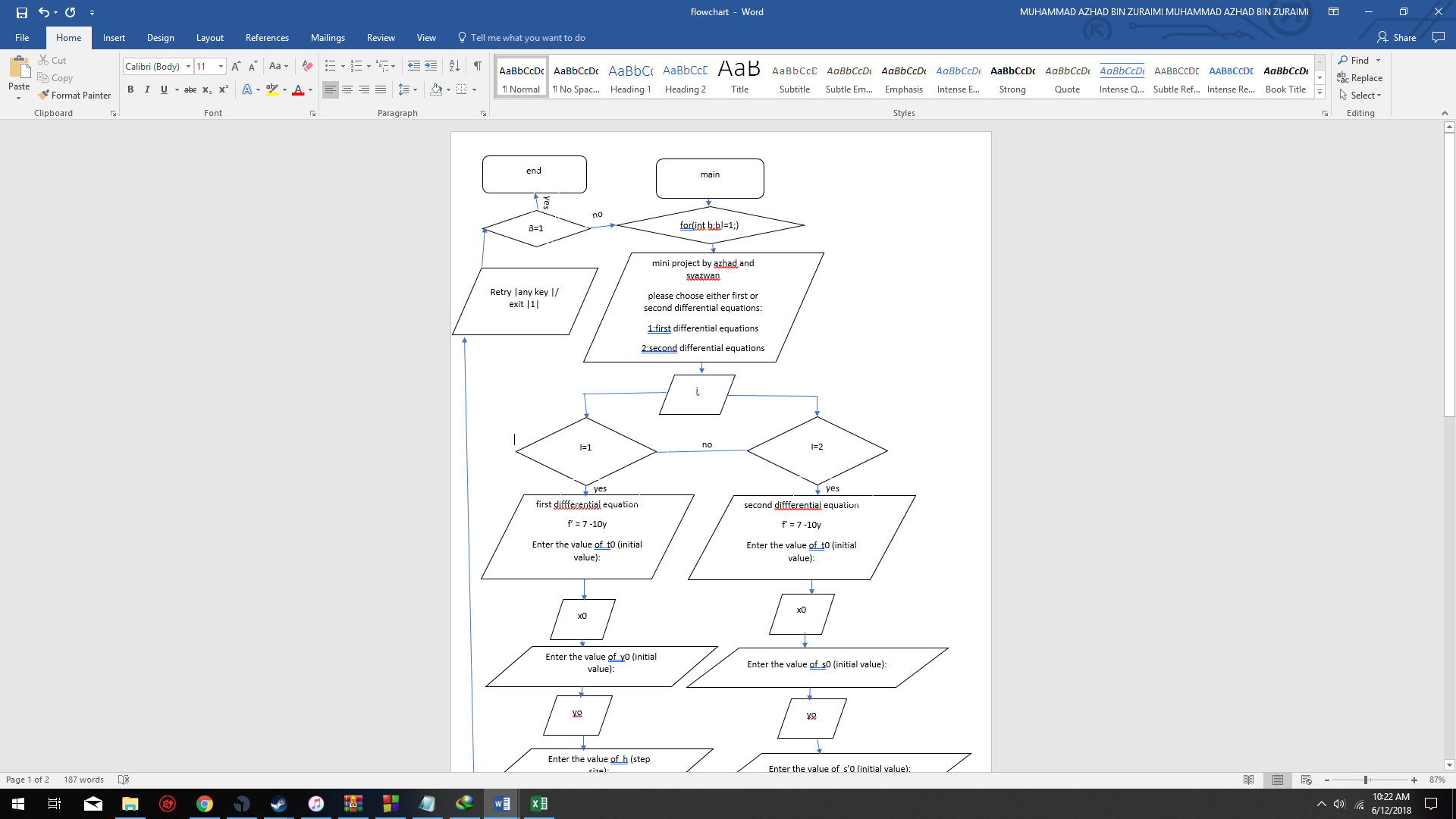
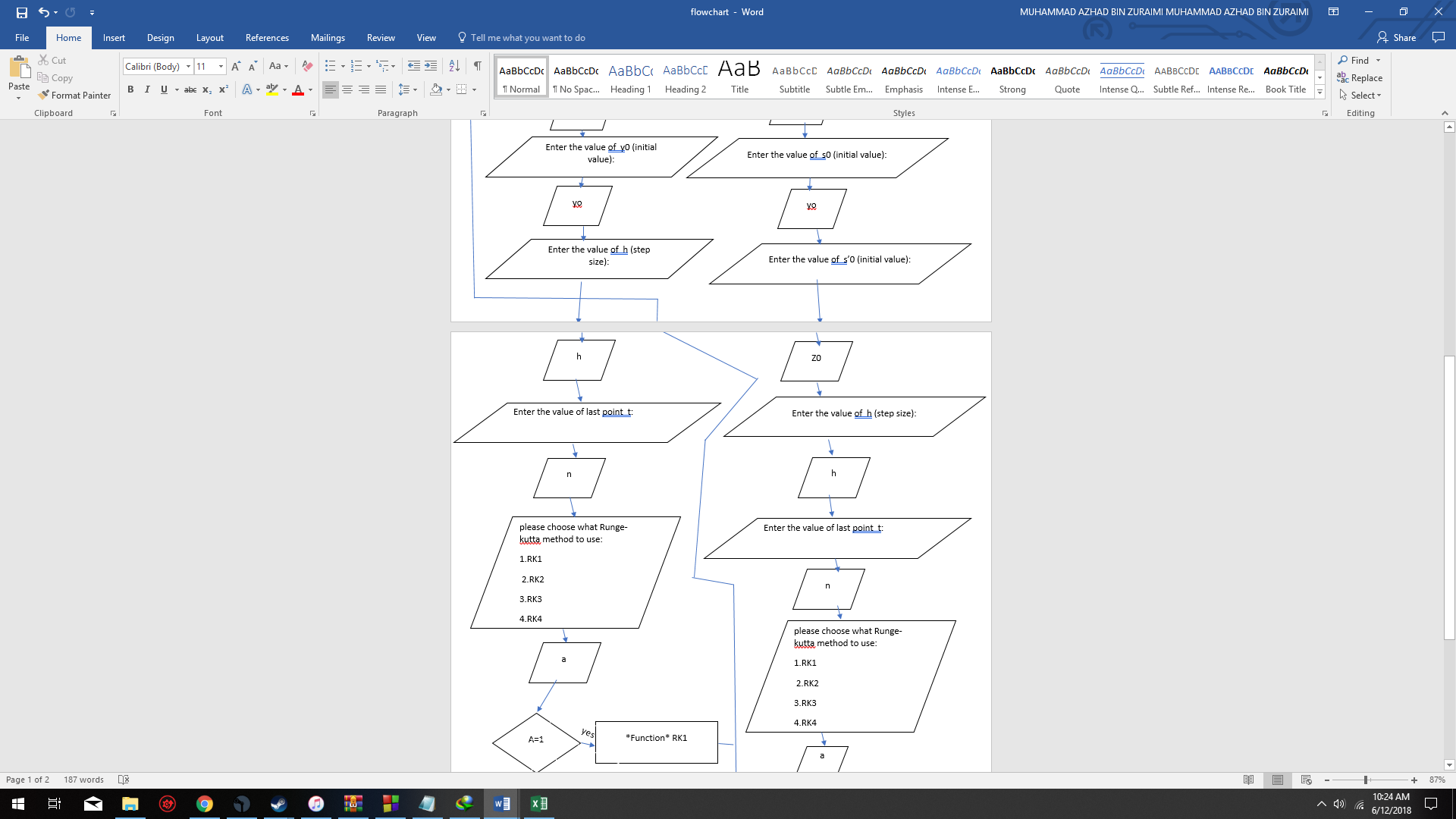
Runge-Kutta is a method of numerically integrating ordinary differential equations by using a trial step at the midpoint of an interval to cancel out lower-order error terms. The method is chosen because it is one of the easiest to solve problems related to First and Second Order Differential Equation. The second-order formula is

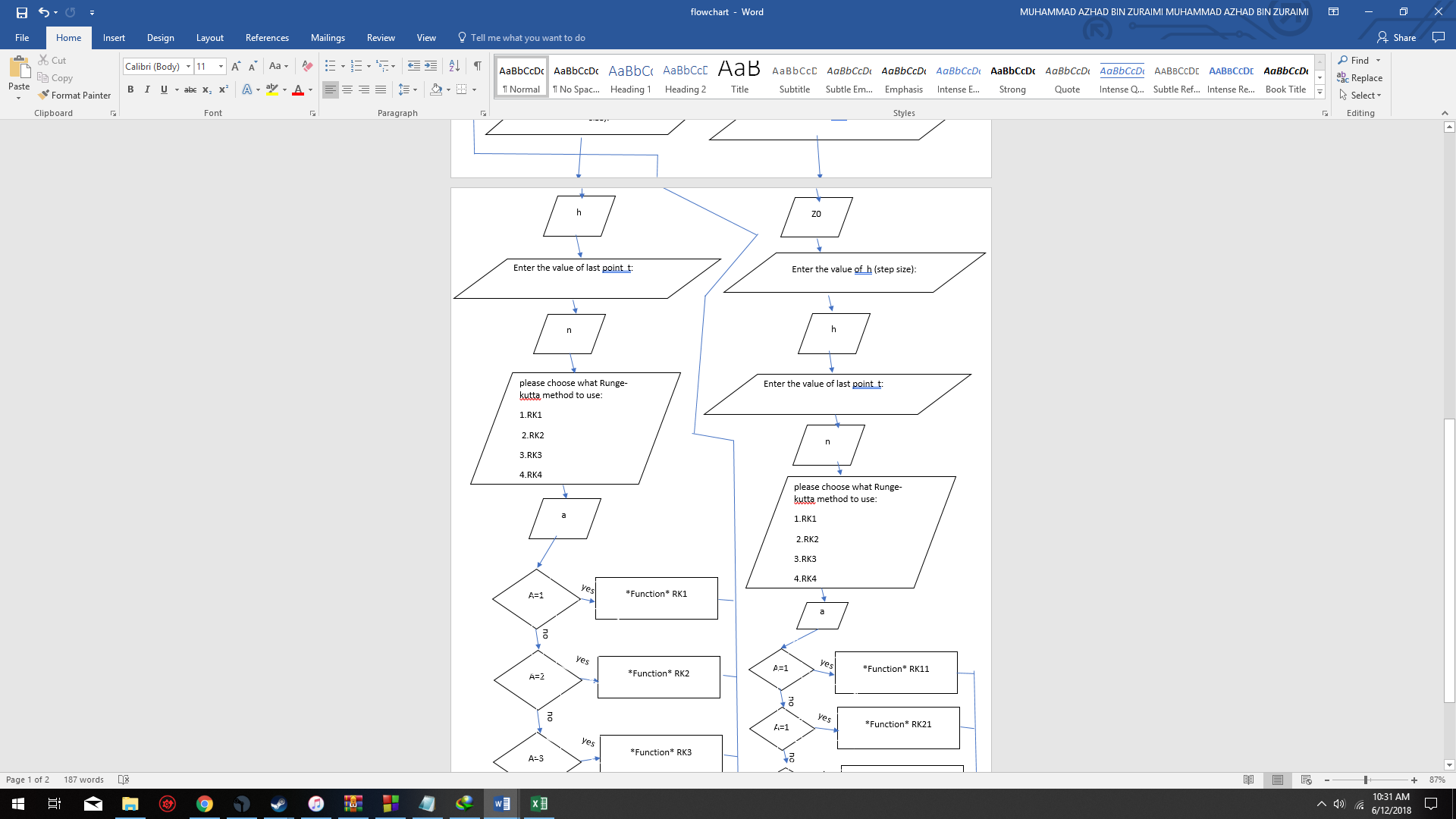
|  |  |  |  |
| --- | --- | --- | --- |
| k_1 | = | hf(x_n,y_n) | (1) |
| k_2 | = | hf(x_n+1/2h,y_n+1/2k_1) | (2) |
| y_(n+1) | = | y_n+k_2+O(h^3) | (3) |

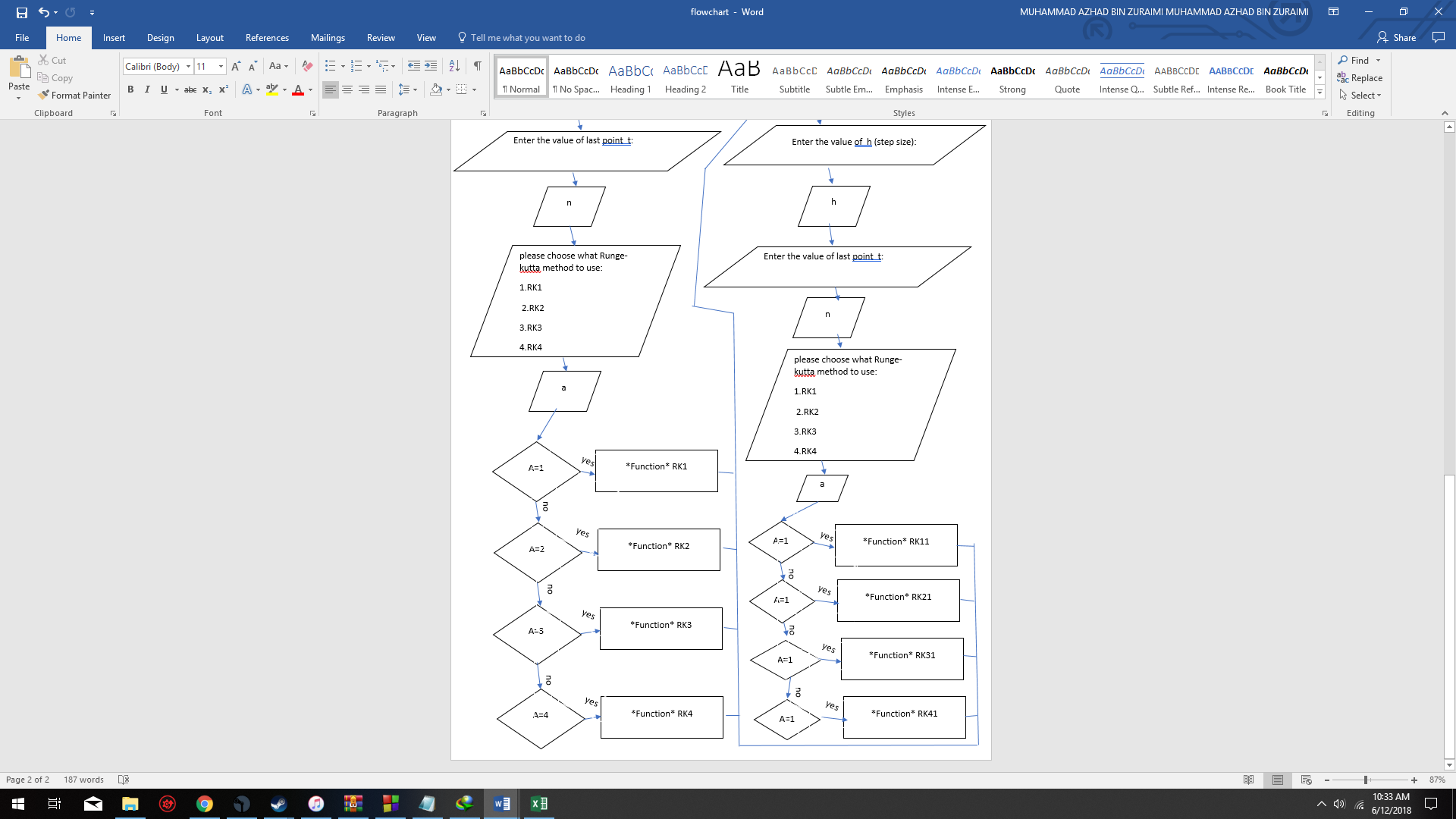
(where O(x) is a Landau symbol), sometimes known as RK2, and the fourth-order formula is

|  |  |  |  |
| --- | --- | --- | --- |
| k_1 | = | hf(x_n,y_n) | (4) |
| k_2 | = | hf(x_n+1/2h,y_n+1/2k_1) | (5) |
| k_3 | = | hf(x_n+1/2h,y_n+1/2k_2) | (6) |
| k_4 | = | hf(x_n+h,y_n+k_3) | (7) |
| y_(n+1) | = | y_n+1/6k_1+1/3k_2+1/3k_3+1/6k_4+O(h^5) | (8) |

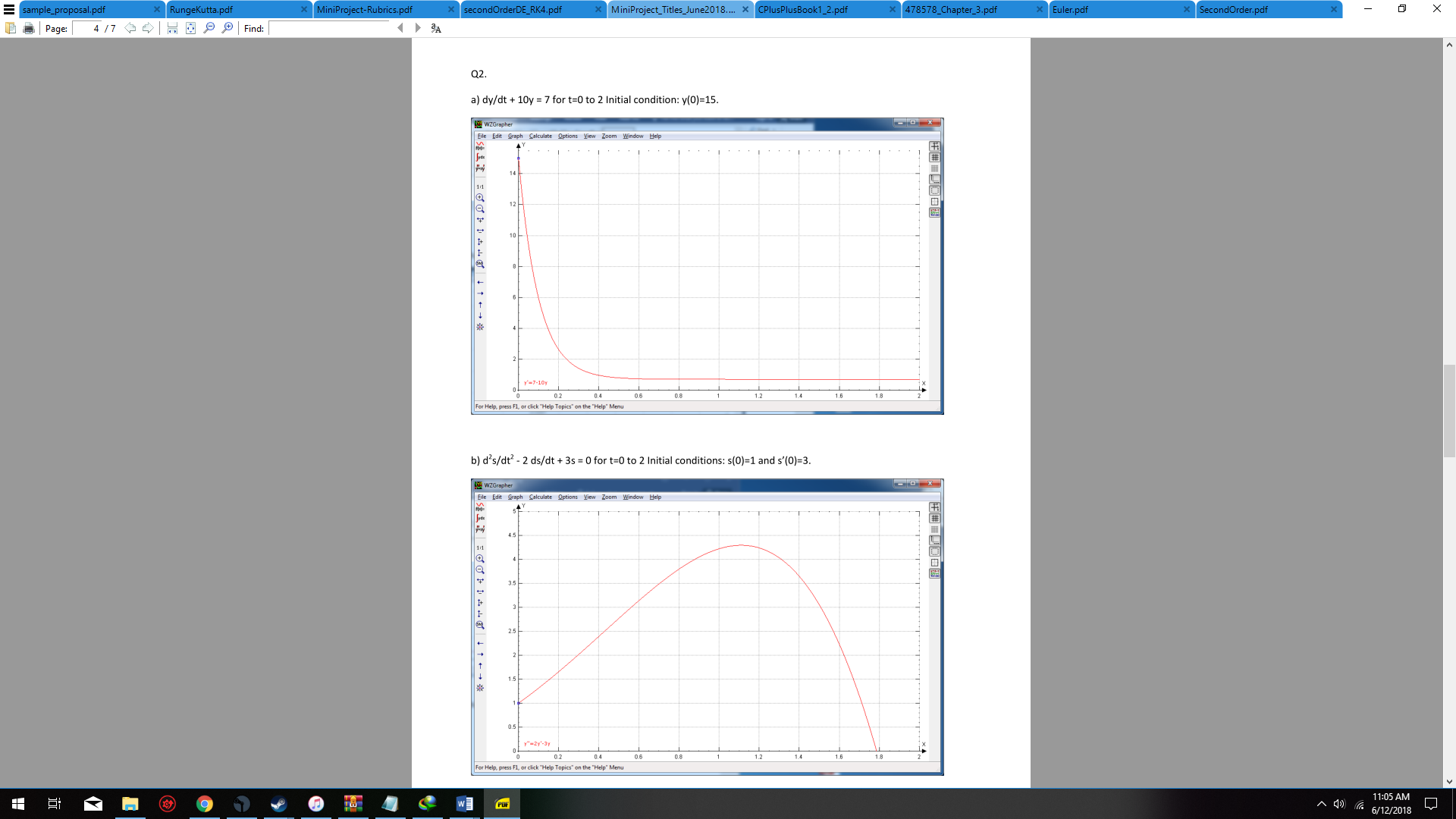
(Press et al. 1992), sometimes known as RK4. This method is reasonably simple and robust and is a good general candidate for numerical solution of differential equations when combined with an intelligent adaptive step-size routine.

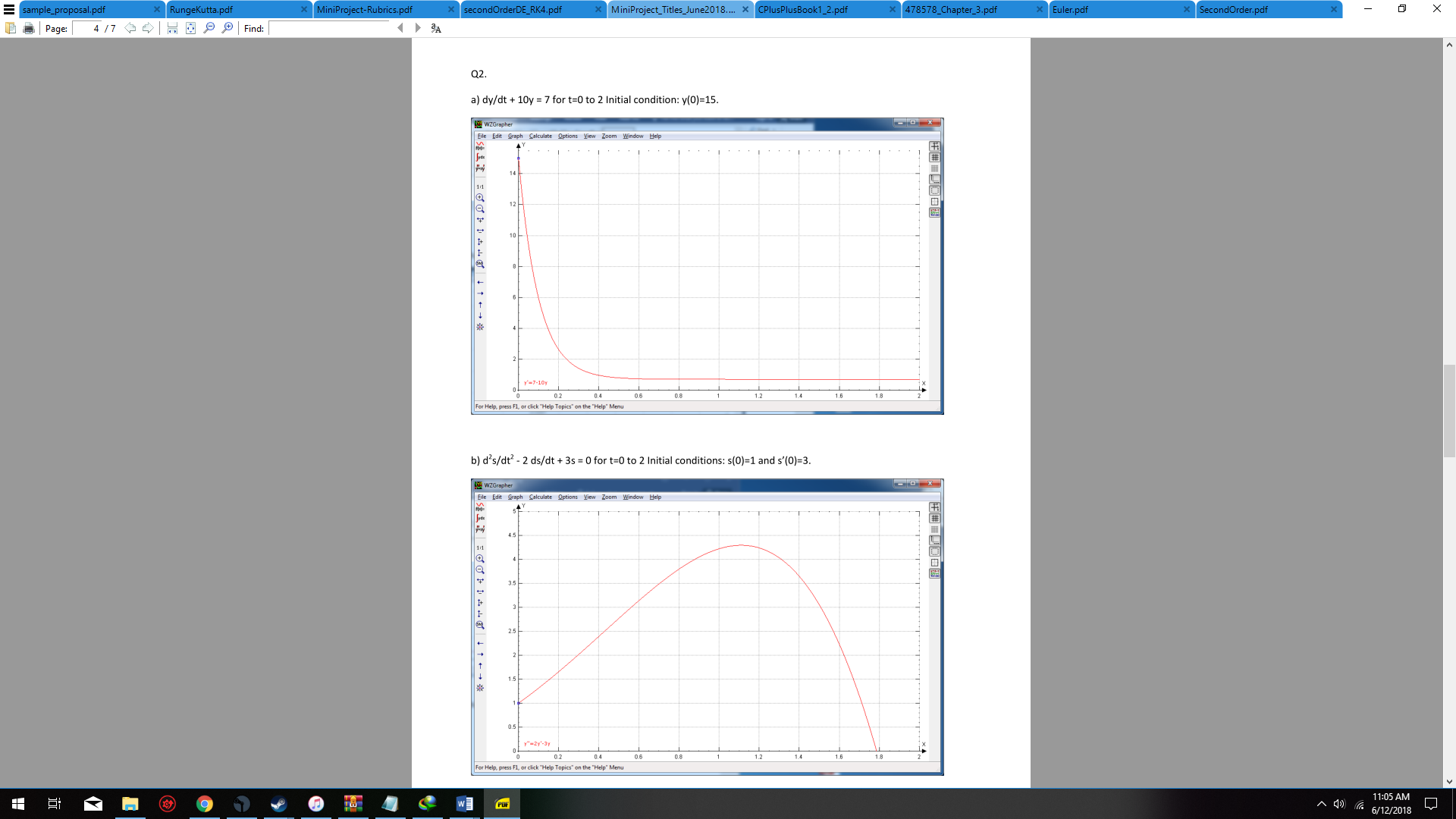
* 1. **Flowchart/Pseudo Code**





* 1. **Validation**





* 1. **Program code**

#include<iostream>

#include <cmath>

#include <fstream>

using namespace std;

void RK1(float xO, float y0,float h,float n);

void RK2(float x0, float y0,float h,float n);

void RK3(float x0, float y0,float h,float n);

void RK4(float x0, float y0,float h,float n);

void RK11(float xO, float y0,float h,float n, float z0 );

void RK21(float xO, float y0,float h,float n, float z0 );

void RK31(float xO, float y0,float h,float n, float z0 );

void RK41(float xO, float y0,float h,float n, float z0 ); //declare functions

class Formula { // classname

private:

float z;

// Data members (variables)

public:

float F(float x, float y);

float I(float x, float y,float z);

float J(float x, float y,float z); //member function to return area

};

float Formula :: F(float x, float y){ //first differential equation

return (7 - 10\*y);

}

float Formula :: I(float x, float y,float z){//second differential equation for y''

return (2\*z-3\*y);

}

float Formula :: J(float x, float y,float z){//second differential equation for y'

return (z);

}

int main() { //main program

cout.precision(2); //precision 2

float y0,x0,z0,n,h;

int i,a;

for( char b;b!=1;){

cout<<"mini project by azhad and syazwan\n";

cout<<"please choose either first or second differential equations:\n";

cout <<"\n1:first order differential equations \n2:second order differential equations\n";

cin >> i;

switch (i) {

case 1: cout <<"\n------------------ \n";

cout <<"\nfirst diffferential equationas\n";

cout <<"\n------------------ \n";

cout <<"\nthe equation is f' = 7 - 10\*y\n";

cout<<"\nEnter the value of t0 (initial value): ";

cin>>x0;

cout<<"\nEnter the value of y0 (initial value): ";

cin>>y0;

cout<<"\nEnter the value of h (step size): ";

cin>>h;

cout<<"\nEnter the value of last point of t: ";

cin>>n;

cout<<"please choose what Runge-kutta method to use:";

cout <<"\n1:RK1 \n2:RK2 \n3:RK3 \n4:RK4\n";

cin>>a;

switch (a){

case 1: RK1(x0,y0,h,n);

break;

case 2: RK2(x0,y0,h,n);

break;

case 3: RK3(x0,y0,h,n);

break;

case 4: RK4(x0,y0,h,n);

break;}

break;

case 2: cout <<"\n------------------ \n";

cout <<"\nsecond diffferential equationas\n";

cout <<"\n------------------ \n";

cout<<"\nEnter the value of t0 (initial value): ";

cin>>x0;

cout<<"\nEnter the value of s0 (initial value): ";

cin>>y0;

cout<<"\nEnter the value of s'0 (initial value): ";

cin>>z0;

cout<<"\nEnter the value of h (step size): ";

cin>>h;

cout<<"\nEnter the value of last point of t: ";

cin>>n;

cout<<"please choose what Runge-kutta method to use:";

cout <<"\n1:RK1 \n2:RK2 \n3:RK3 \n4:RK4\n";

cin>>a;

switch (a){

case 1: RK11(x0,y0,h,n,z0);

break;

case 2: RK21(x0,y0,h,n,z0);

break;

case 3: RK31(x0,y0,h,n,z0);

break;

case 4: RK41(x0,y0,h,n,z0);

break;}

break;

}

cout<<"\n\nretry? |any key| / exit |1| \n\n"; //retry

cin>>b;}}

void RK1(float x0, float y0,float h,float n){//first-order Runge-kutta method in first differential equation

ofstream kutta ("Runge-kutta.txt");

float y1,f,k[1];

Formula form;

cout<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

kutta<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl;

for( ; x0<n; x0=x0+h){

f=form.F(x0,y0);

k[0] = h \* f;

y1 = y0 +k[0];

cout<<"\n k1 = "<<k[0];

kutta<<"\n k1 = "<<k[0];

cout<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

kutta<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

y0=y1;

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl;

}} //take new point for next calculation }

void RK2(float x0, float y0,float h,float n){//second-order Runge-kutta method in first differential equation

ofstream kutta ("Runge-kutta.txt");

float y1,f,k[2];

Formula form;

cout<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

kutta<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl;

for( ; x0<n; x0=x0+h){

f=form.F(x0,y0);

k[0] = h \* f;

f=form.F(x0+2\*h/3,y0+2\*k[0]/3);

k[1] = h \* f;

y1 = y0 + (k[0]+3\*k[1])/4; //new point

cout<<"\n k1 = "<<k[0];

kutta<<"\n k1 = "<<k[0];

cout<<"\n k2 = "<<k[1];

kutta<<"\n k2 = "<<k[1];

cout<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

kutta<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

y0=y1;

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl; //take new point for next calculation

}

}

void RK3(float x0, float y0,float h,float n){//third-order Runge-kutta method in first differential equation

ofstream kutta ("Runge-kutta.txt");

float y1,f,k[3];

Formula form;

cout<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

kutta<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl;

for( ; x0<n; x0=x0+h){

f = form.F(x0,y0);

k[0] = h \* f;

f = form.F(x0+h/2,y0+k[0]/2);

k[1] = h \* f;

f = form.F(x0+h,y0+2\*k[1]-k[0]);

k[2] = h\*f;

y1 = y0 + ( k[0] + 4\*k[1] + k[2])/6; //new point

cout<<"\n k1 = "<<k[0];

kutta<<"\n k1 = "<<k[0];

cout<<"\n k2 = "<<k[1];

kutta<<"\n k2 = "<<k[1];

cout<<"\n k3 = "<<k[2];

kutta<<"\n k3 = "<<k[2];

cout<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

kutta<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

y0=y1;

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl; //take new point for next calculation

}

}

void RK4(float x0, float y0,float h,float n){//fourth-order Runge-kutta method in first differential equation

ofstream kutta ("Runge-kutta.txt");

float y1,f,k[4];

Formula form;

cout<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

kutta<<"\nStart point: \t\t\tHistogram\n\n x = "<<x0<<" y = "<<y0<<"\t\t\t";

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl;

for( ; x0<n; x0=x0+h){

f = form.F(x0,y0);

k[0] = h \* f;

f = form.F(x0+h/2,y0+k[0]/2);

k[1] = h \* f;

f = form.F(x0+h/2,y0+k[1]/2);

k[2] = h \* f;

f = form.F(x0+h,y0+k[2]);

k[3] = h \* f;

y1 = y0 + ( k[0] + 2\*k[1] + 2\*k[2] + k[3])/6; //new point

cout<<"\n k1 = "<<k[0];

kutta<<"\n k1 = "<<k[0];

cout<<"\n k2 = "<<k[1];

kutta<<"\n k2 = "<<k[1];

cout<<"\n k3 = "<<k[2];

kutta<<"\n k3 = "<<k[2];

cout<<"\n k4 = "<<k[3];

kutta<<"\n k4 = "<<k[3];

cout<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

kutta<<"\n x = "<<x0+h<<" y = "<<y1<<"\n\t\t\t\t";

y0=y1;

for (float a=0.1; a<=y0; a+=0.1){

cout<<("\*");

kutta<<("\*");

}

cout<<endl;

kutta<<endl; //take new point for next calculation

}

}

void RK11(float x0, float y0,float h,float n, float z0){//first-order Runge-kutta method in second differential equation

float y1,z1,i1,j1;

Formula form;

cout<<"\nStart point: \n\n x = "<<x0<<" and y = "<<y0<<endl;

for( ; x0<n; x0=x0+h){

j1=h\*form.J(x0,y0,z0);

i1=h\*form.I(x0,y0,z0);

y1 = h\*i1;

z1 = h\*j1;

cout<<"\n i1 = "<<i1;

cout<<"\n j1 = "<<j1;

cout<<"\n x = "<<x0+h<<" and y = "<<y1<<" and y' = "<<z1<<endl;

y0=y1;

z0=z1; }} //take new point for next calculation }

void RK21(float x0, float y0,float h,float n, float z0){//first-order Runge-kutta method in second differential equation

float y1,z1,i1,i2,j1,j2;

Formula form;

cout<<"\nStart point: \n\n x = "<<x0<<" and y = "<<y0<<endl;

for( ; x0<n; x0=x0+h){

j1=h\*form.J(x0,y0,z0);

i1=h\*form.I(x0,y0,z0);

j2=h\*form.J(x0+h/2,y0+i1/2,z0+i1/2);

i2=h\*form.I(x0+h/2,y0+j1/2,z0+j1/2);

y1 = y0 + i2 ;

z1 = z0 + j2; //new point

cout<<"\n i1 = "<<i1;

cout<<"\n j1 = "<<j1;

cout<<"\n i2 = "<<i2;

cout<<"\n j2 = "<<j2;

cout<<"\n x = "<<x0+h<<" and y = "<<y1<<" and y' = "<<z1<<endl;

y0=y1;

z0=z1; //take new point for next calculation }

}}

void RK31(float x0, float y0,float h,float n, float z0){//first-order Runge-kutta method in second differential equation

float y1,z1,i1,i2,i3,j1,j2,j3;

Formula form;

cout<<"\nStart point: \n\n x = "<<x0<<" and y = "<<y0<<endl;

for( ; x0<n; x0=x0+h){

j1=h\*form.J(x0,y0,z0);

i1=h\*form.I(x0,y0,z0);

j2=h\*form.J(x0+h/2,y0+j1/2,z0+i1/2);

i2=h\*form.I(x0+h/2,y0+j1/2,z0+i1/2);

j3=h\*form.J(x0+h,y0+2\*j2-j1,z0+2\*i2-i1);

i3=h\*form.I(x0+h,y0+j2/2,z0+i2/2);

y1 = y0 + ( i1 + 4\*i2 + i3 )/6;

z1 = z0 + ( j1 + 4\*j2 + j3 )/6; //new point

cout<<"\n i1 = "<<i1;

cout<<"\n j1 = "<<j1;

cout<<"\n i2 = "<<i2;

cout<<"\n j2 = "<<j2;

cout<<"\n i3 = "<<i3;

cout<<"\n j3 = "<<j3;

cout<<"\n x = "<<x0+h<<" and y = "<<y1<<" and y' = "<<z1<<endl;

y0=y1;

z0=z1; //take new point for next calculation }

}}

void RK41(float x0, float y0,float h,float n,float z0){//first-order Runge-kutta method in second differential equation

float y1,z1,i1,i2,i3,i4,j1,j2,j3,j4;

Formula form;

cout<<"\nStart point: \n\n x = "<<x0<<" and y = "<<y0<<endl;

for( ; x0<n; x0=x0+h){

j1=h\*form.J(x0,y0,z0);

i1=h\*form.I(x0,y0,z0);

j2=h\*form.J(x0+h/2,y0+j1/2,z0+i1/2);

i2=h\*form.I(x0+h/2,y0+j1/2,z0+i1/2);

j3=h\*form.J(x0+h/2,y0+j2/2,z0+i2/2);

i3=h\*form.I(x0+h/2,y0+j2/2,z0+i2/2);

j4=h\*form.J(x0+h,y0+j3,z0+i3);

i4=h\*form.I(x0+h,y0+j3,z0+i3);

y1 = y0 + ( i1 + 2\*i2 + 2\*i3 + i4)/6;

z1 = z0 + ( j1 + 2\*j2 + 2\*j3 + j4)/6; //new point

cout<<"\n i1 = "<<i1;

cout<<"\n j1 = "<<j1;

cout<<"\n i2 = "<<i2;

cout<<"\n j2 = "<<j2;

cout<<"\n i3 = "<<i3;

cout<<"\n j3 = "<<j3;

cout<<"\n i4 = "<<i4;

cout<<"\n j4 = "<<i4;

cout<<"\n x = "<<x0+h<<" and y = "<<y1<<" and y' = "<<z1<<endl;

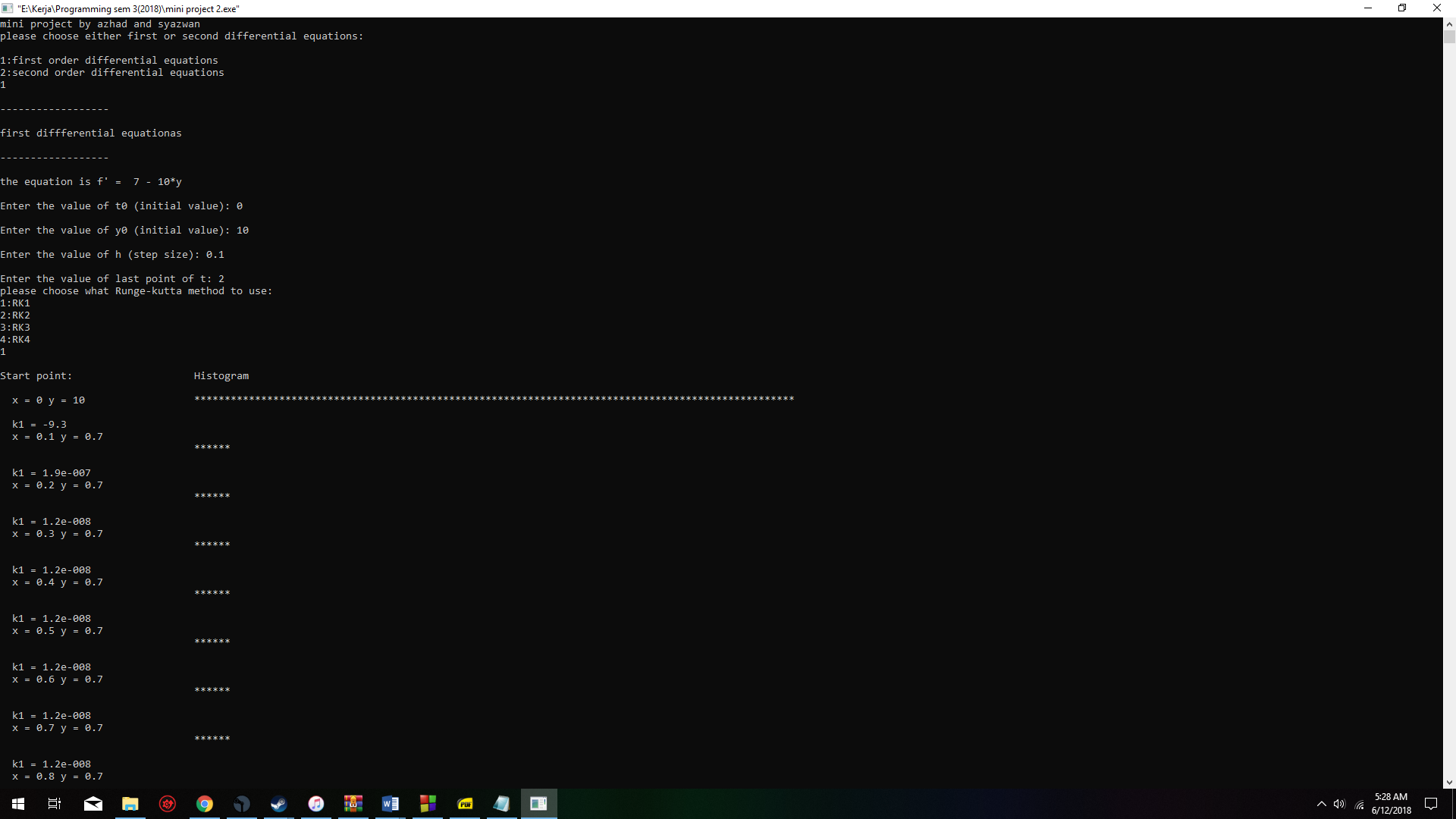
y0=y1;

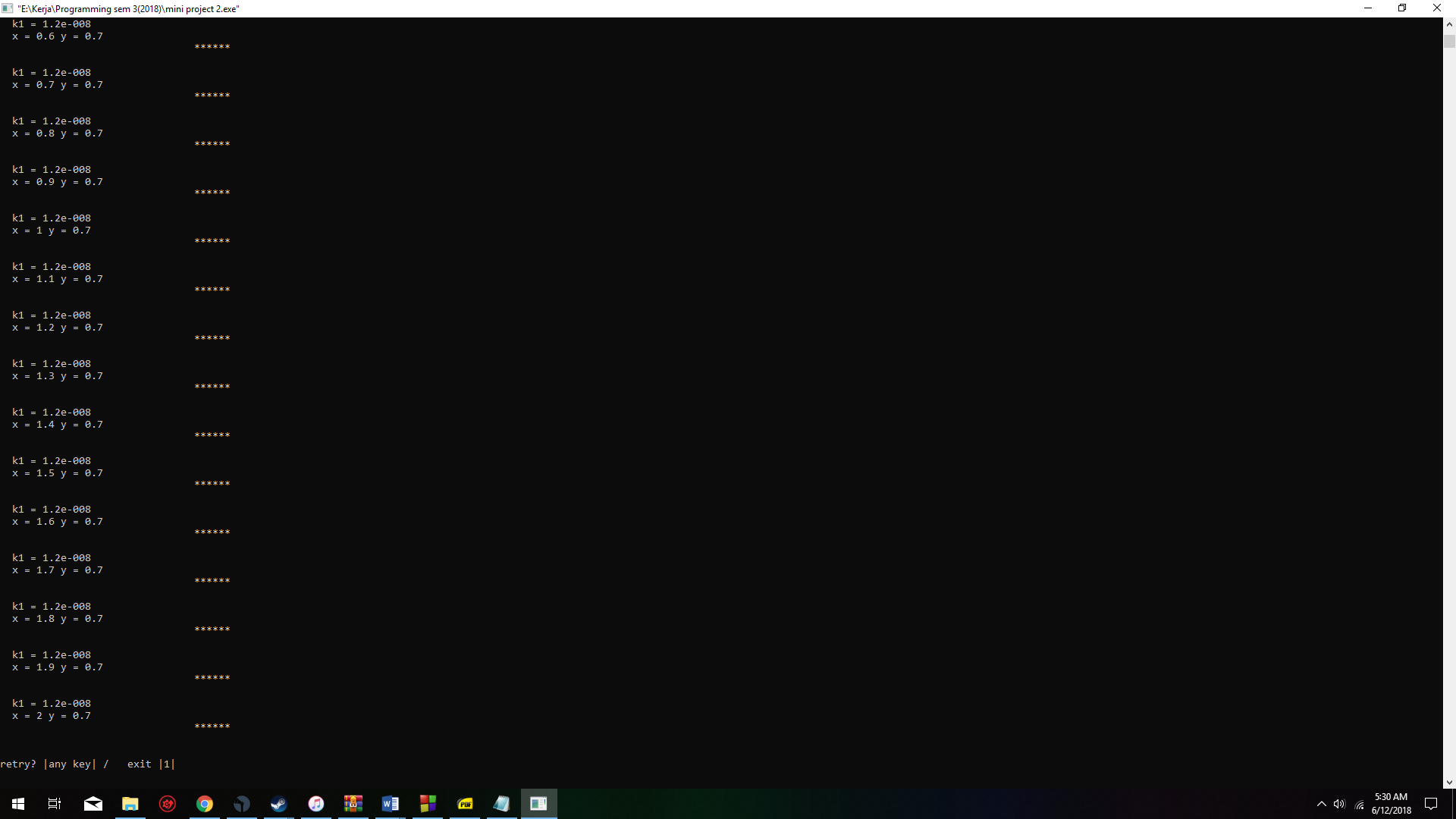
z0=z1; //take new point for next calculation

}}

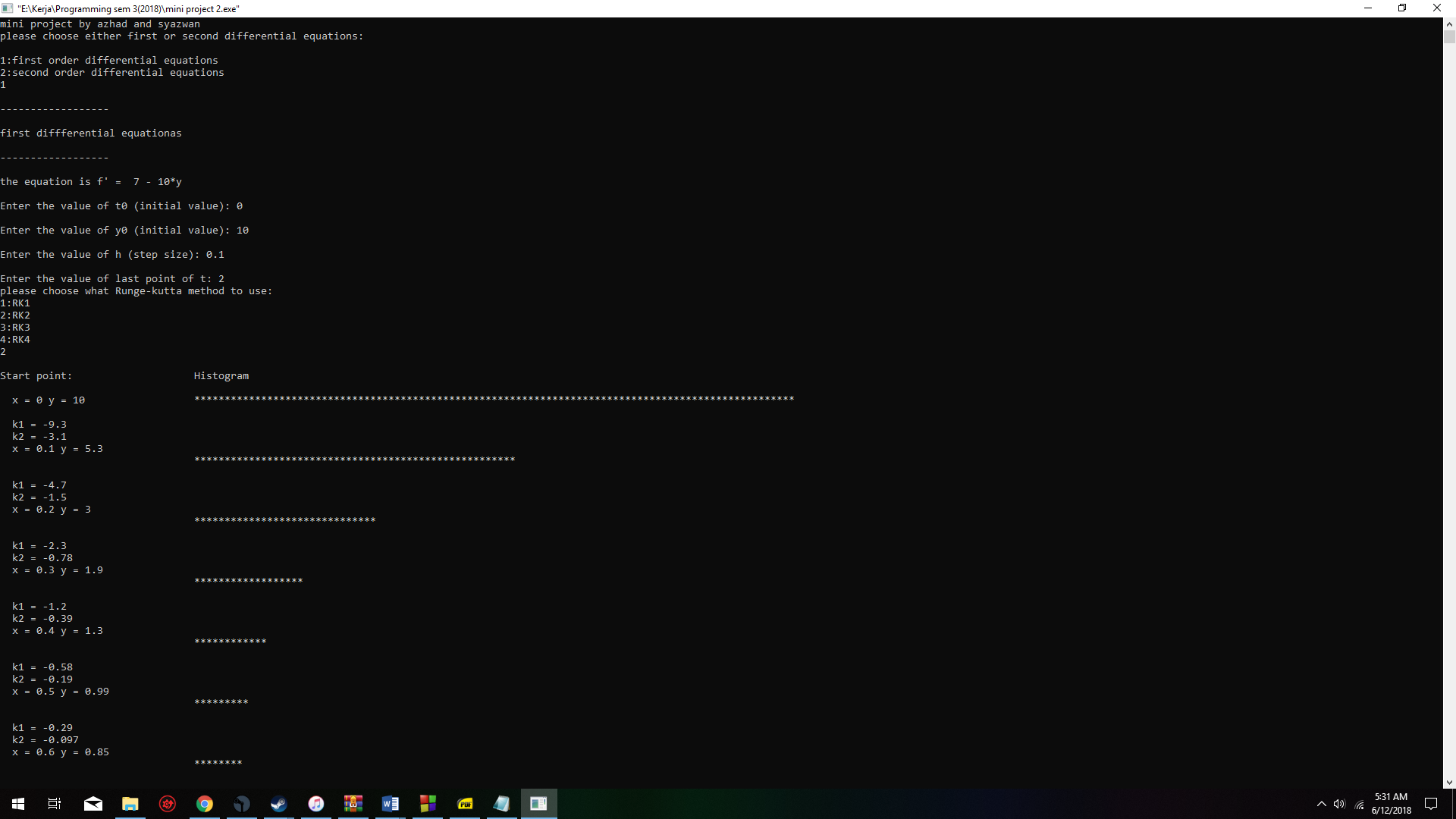
* 1. **Sample Results**

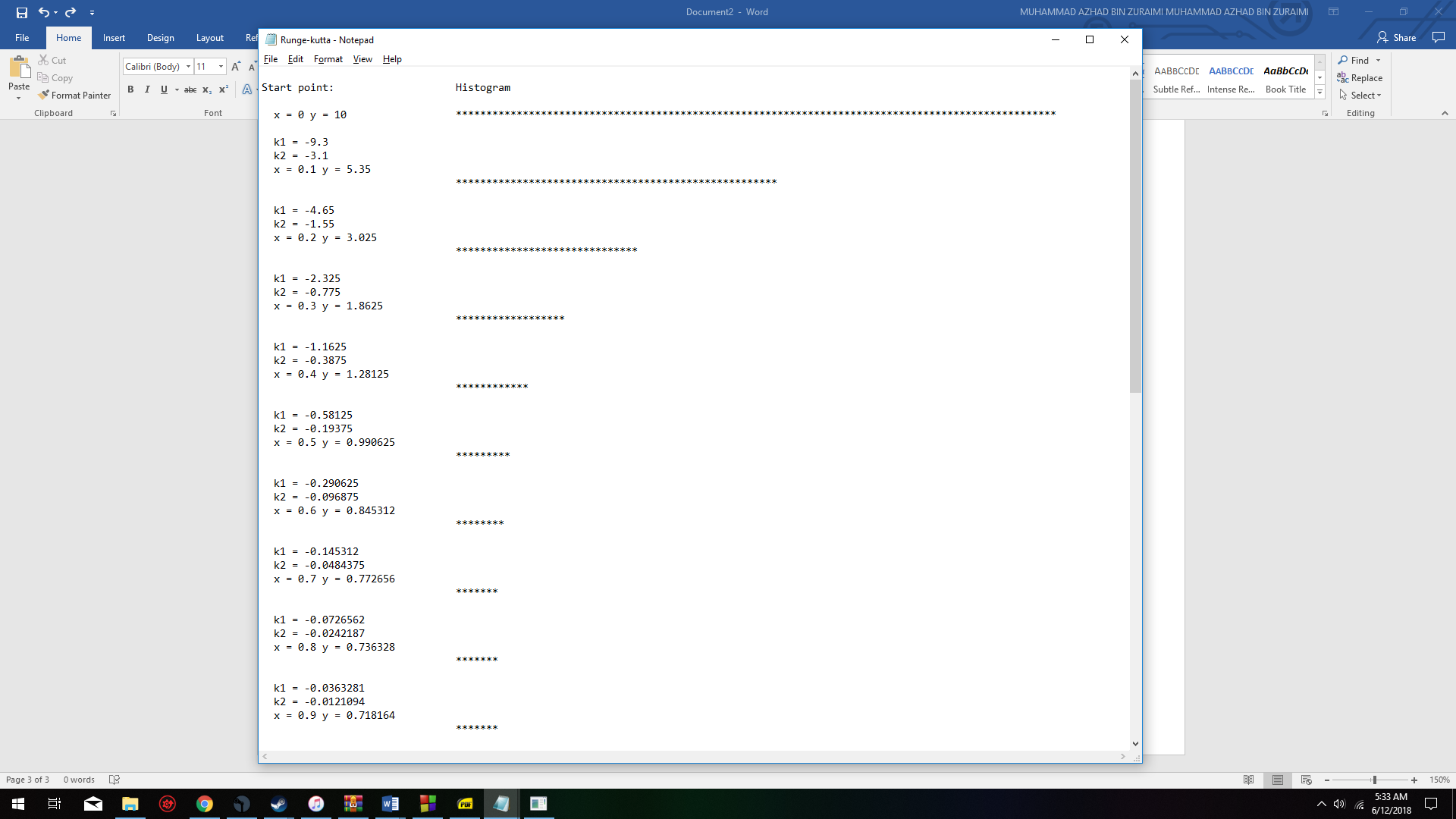
RK1 solve

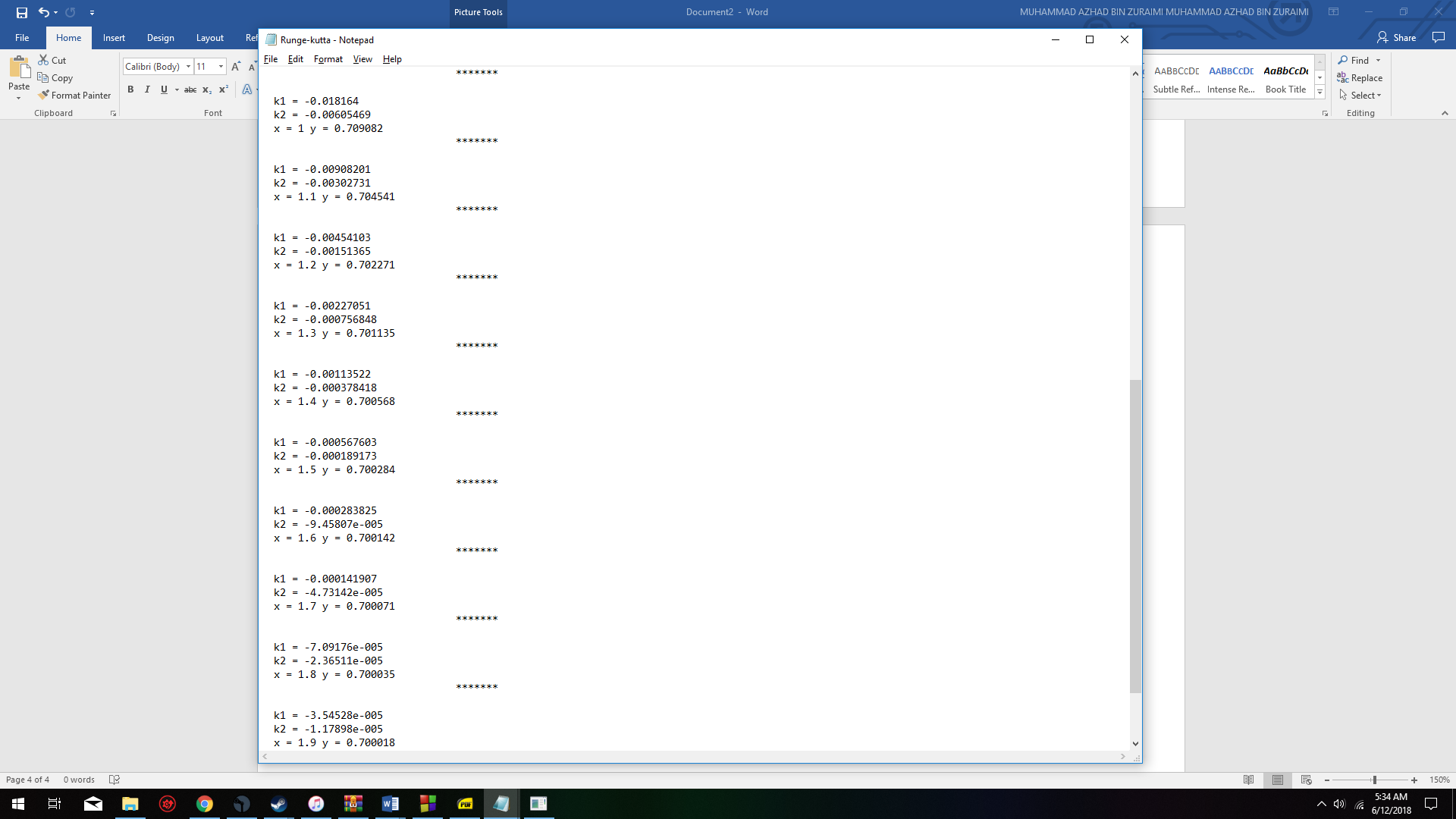


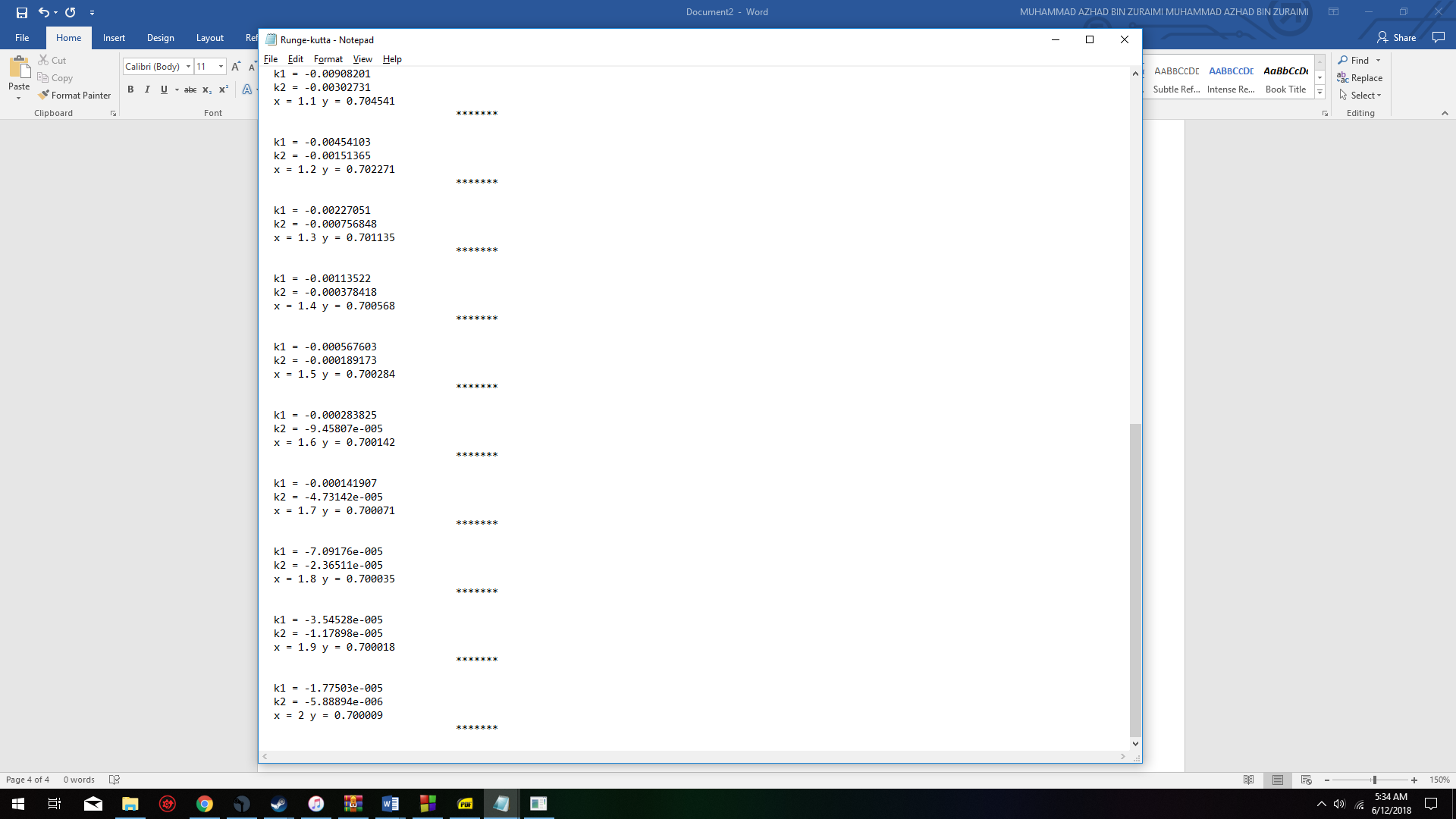


RK2 solve

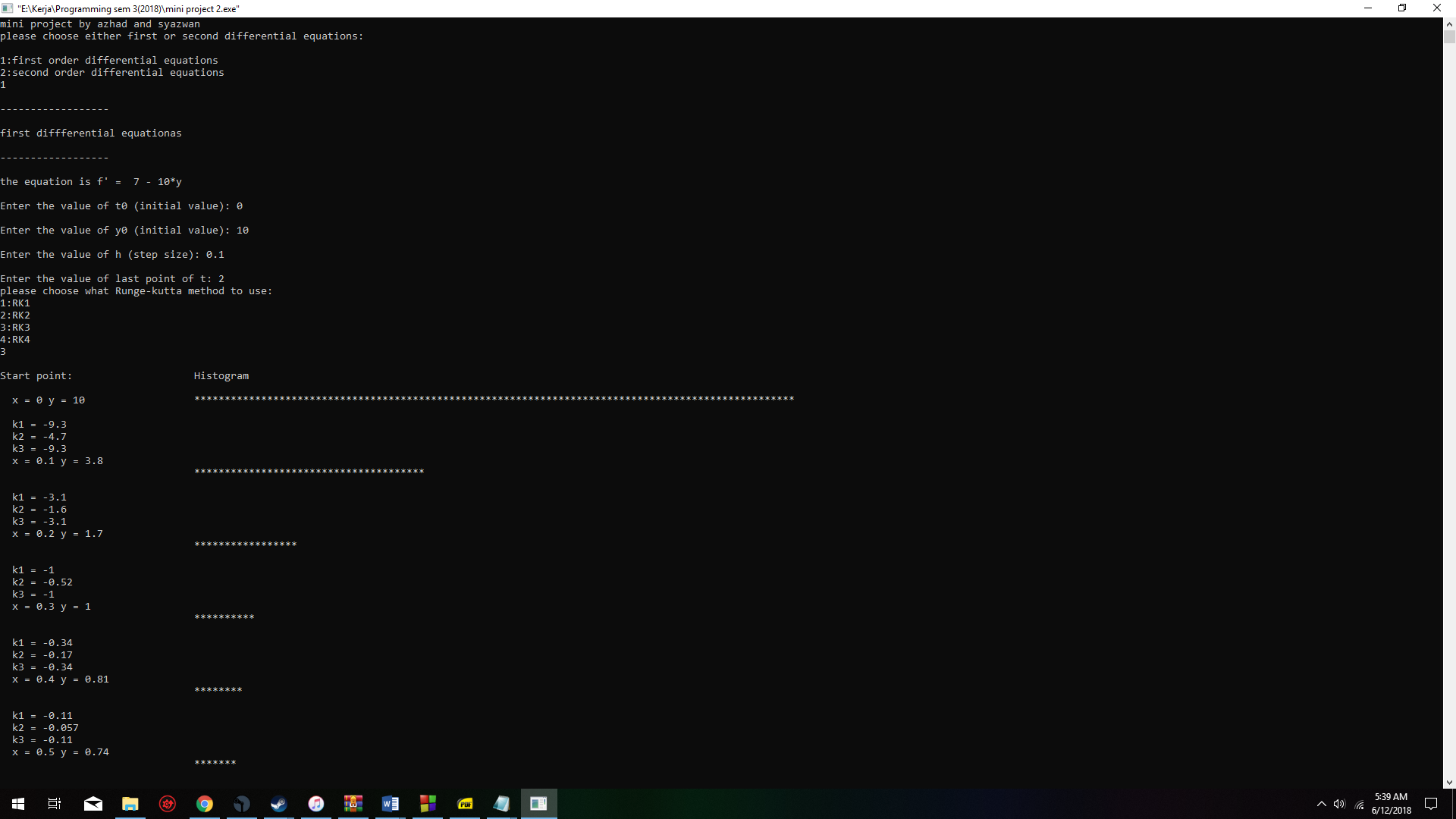


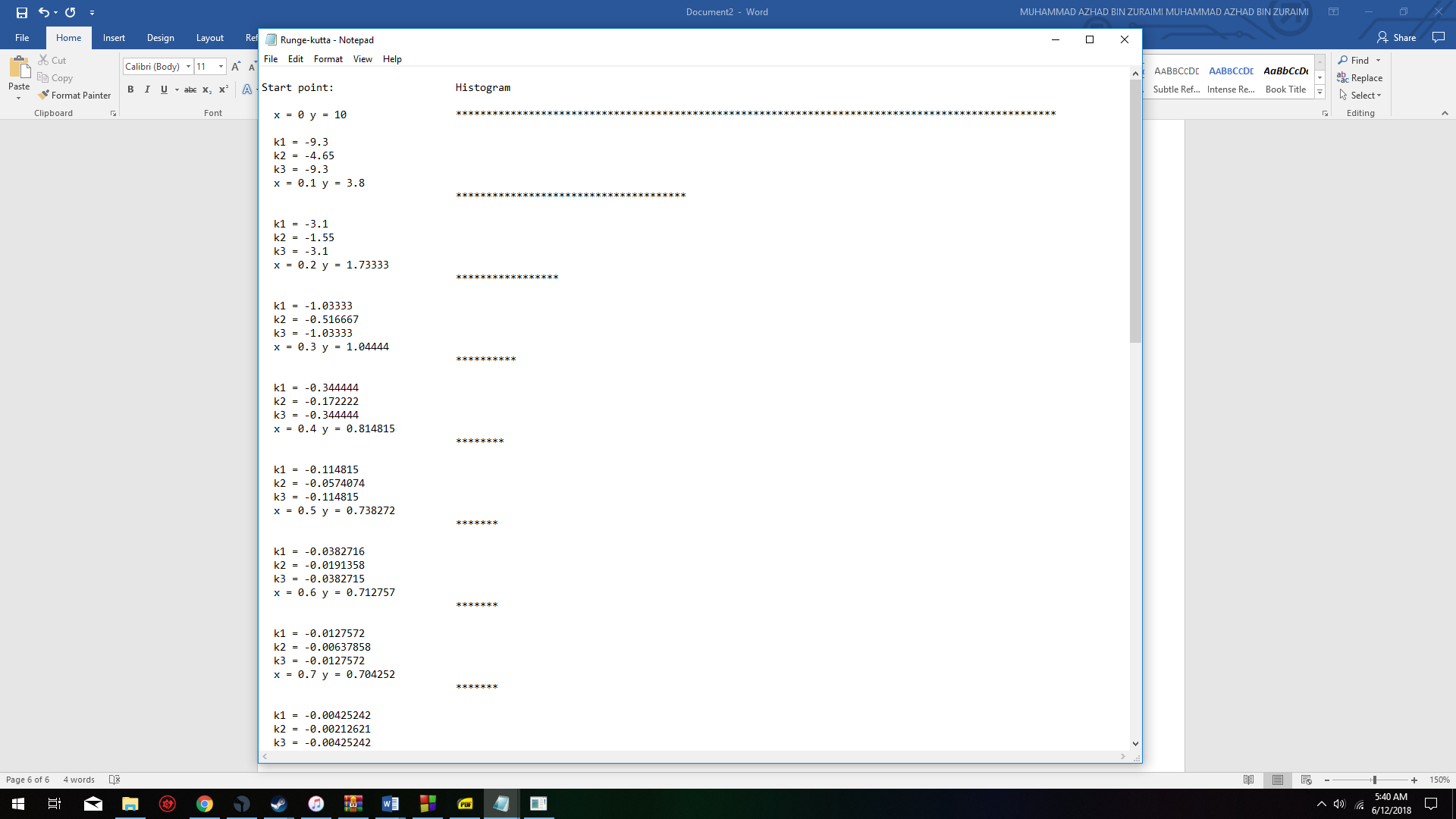


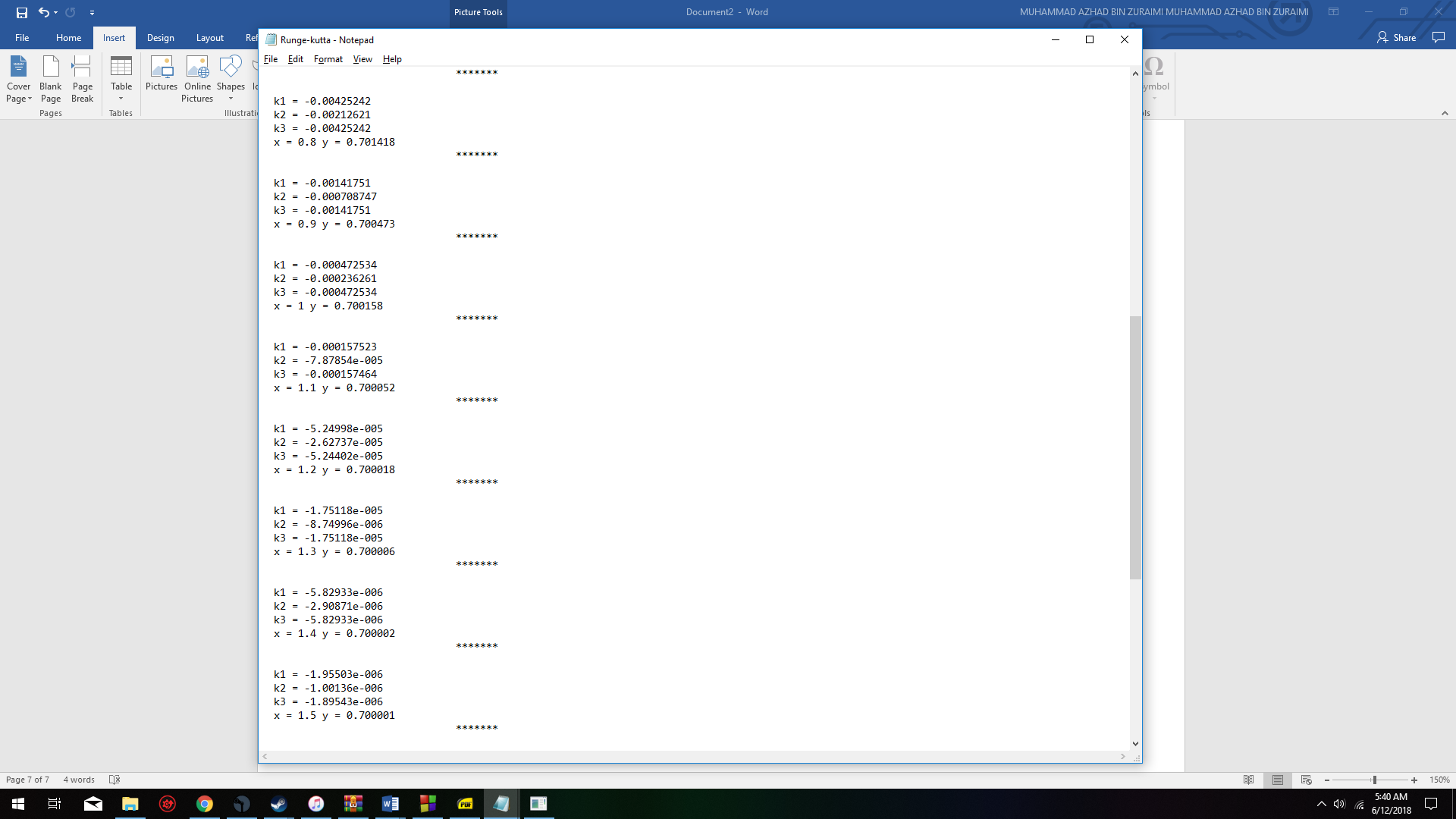


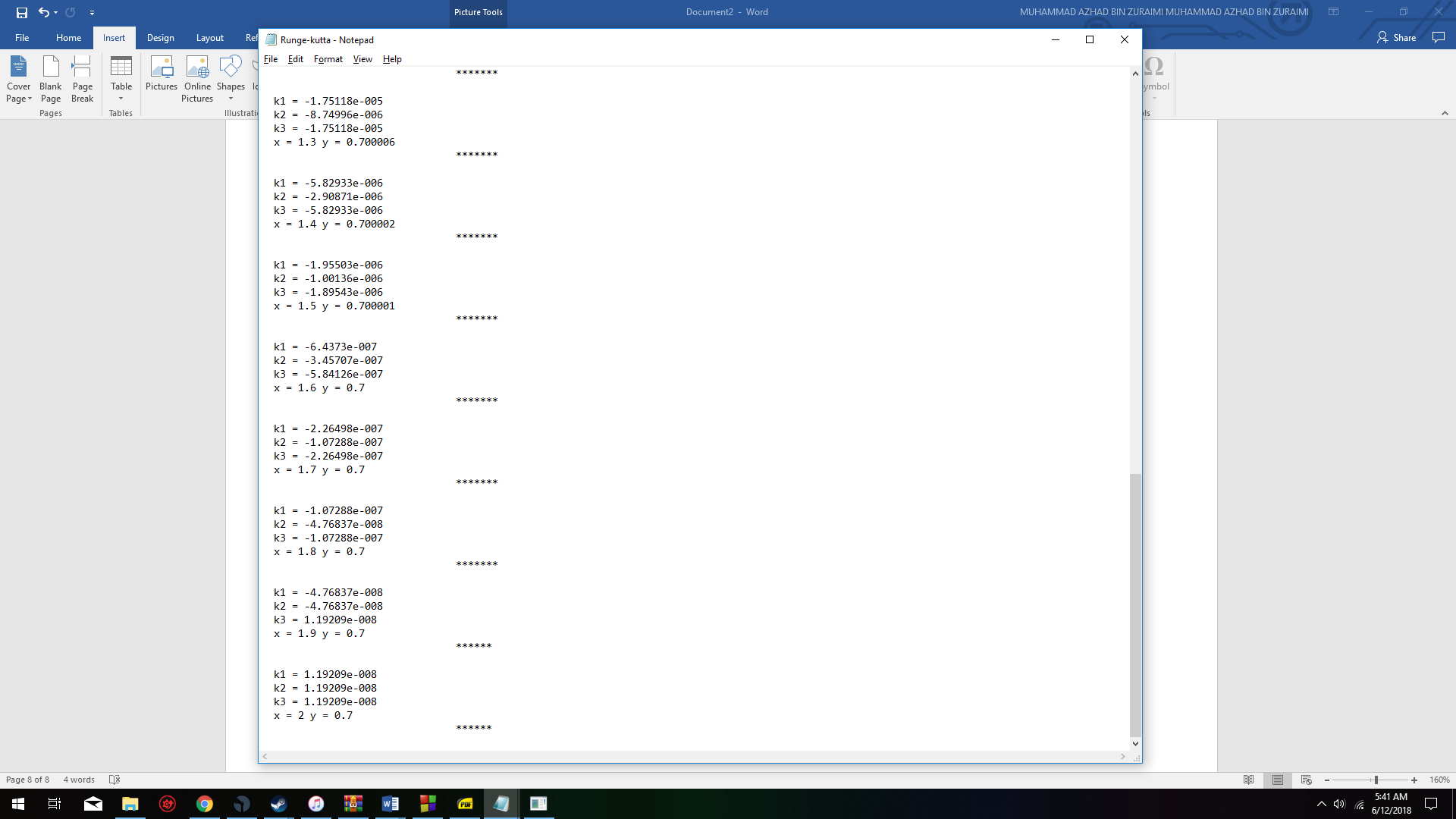


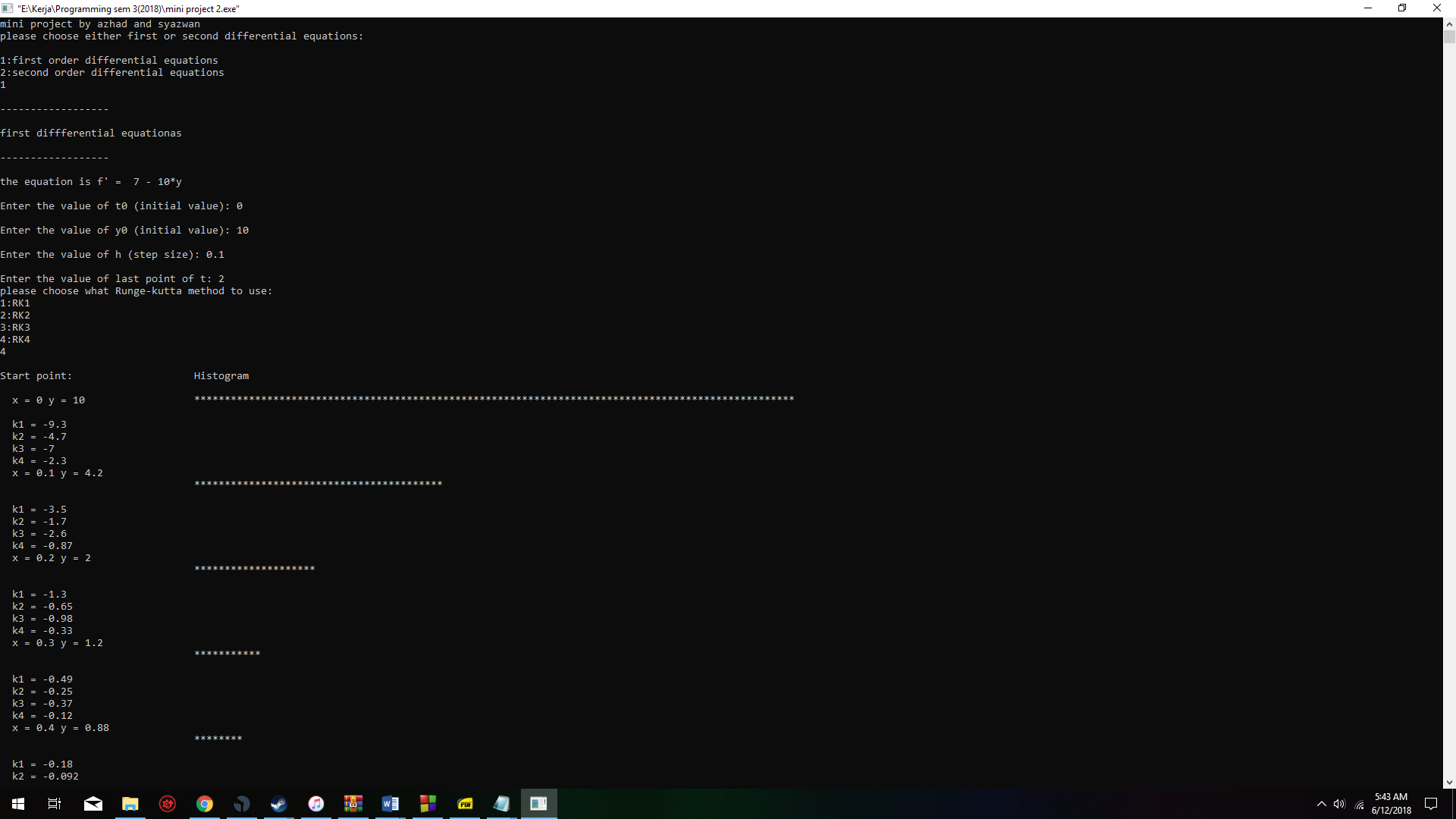
Solve RK 3



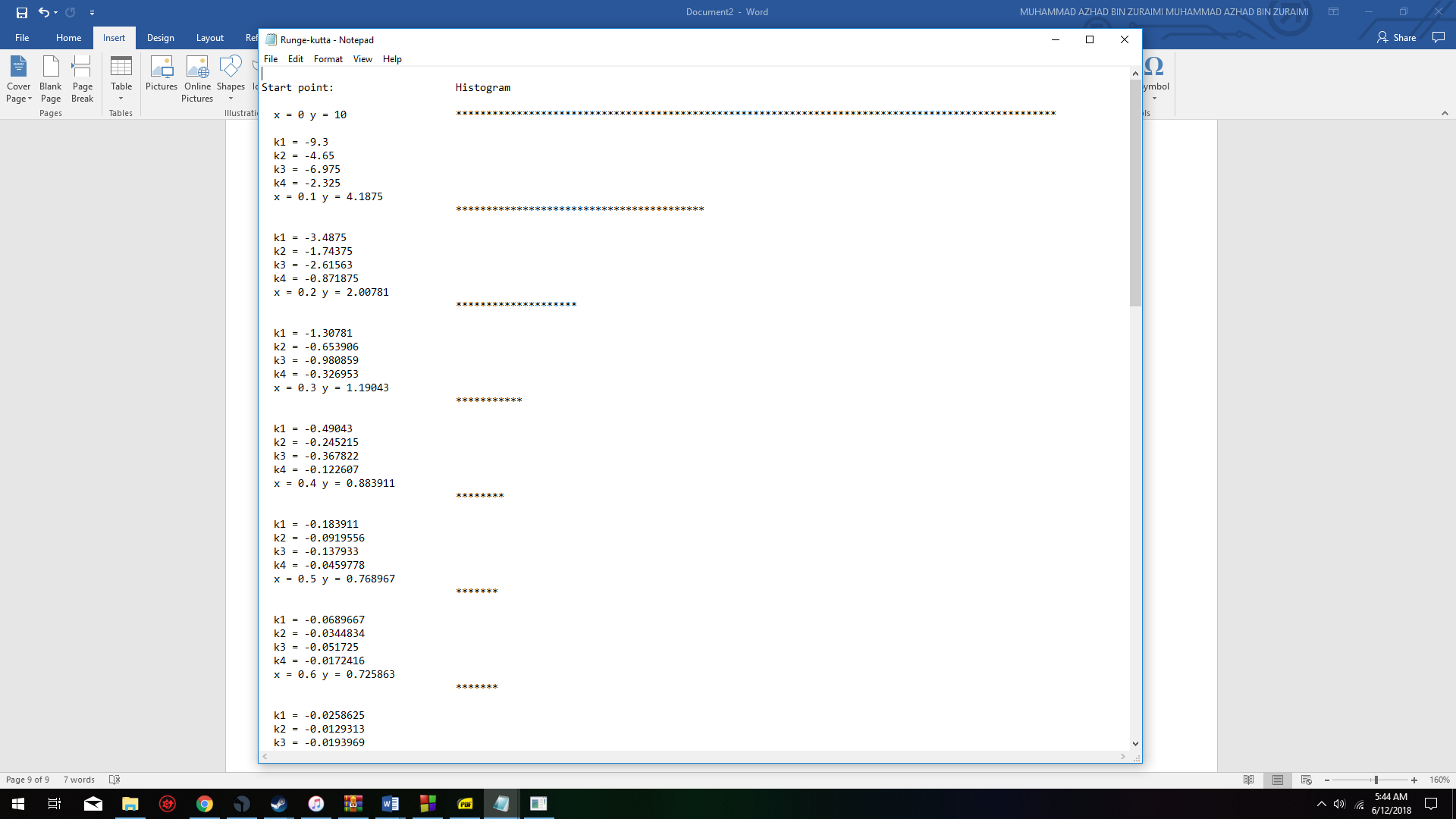


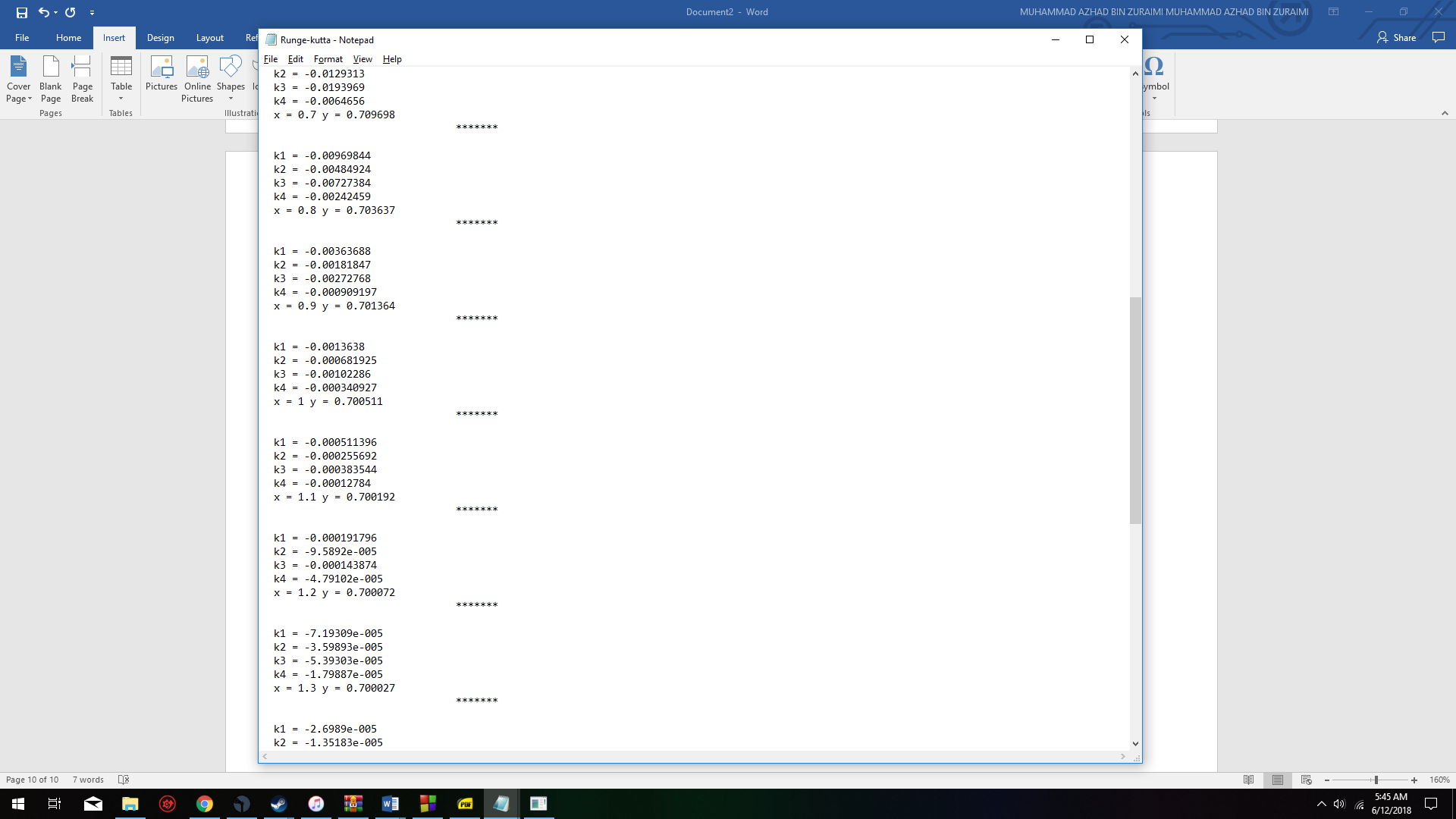


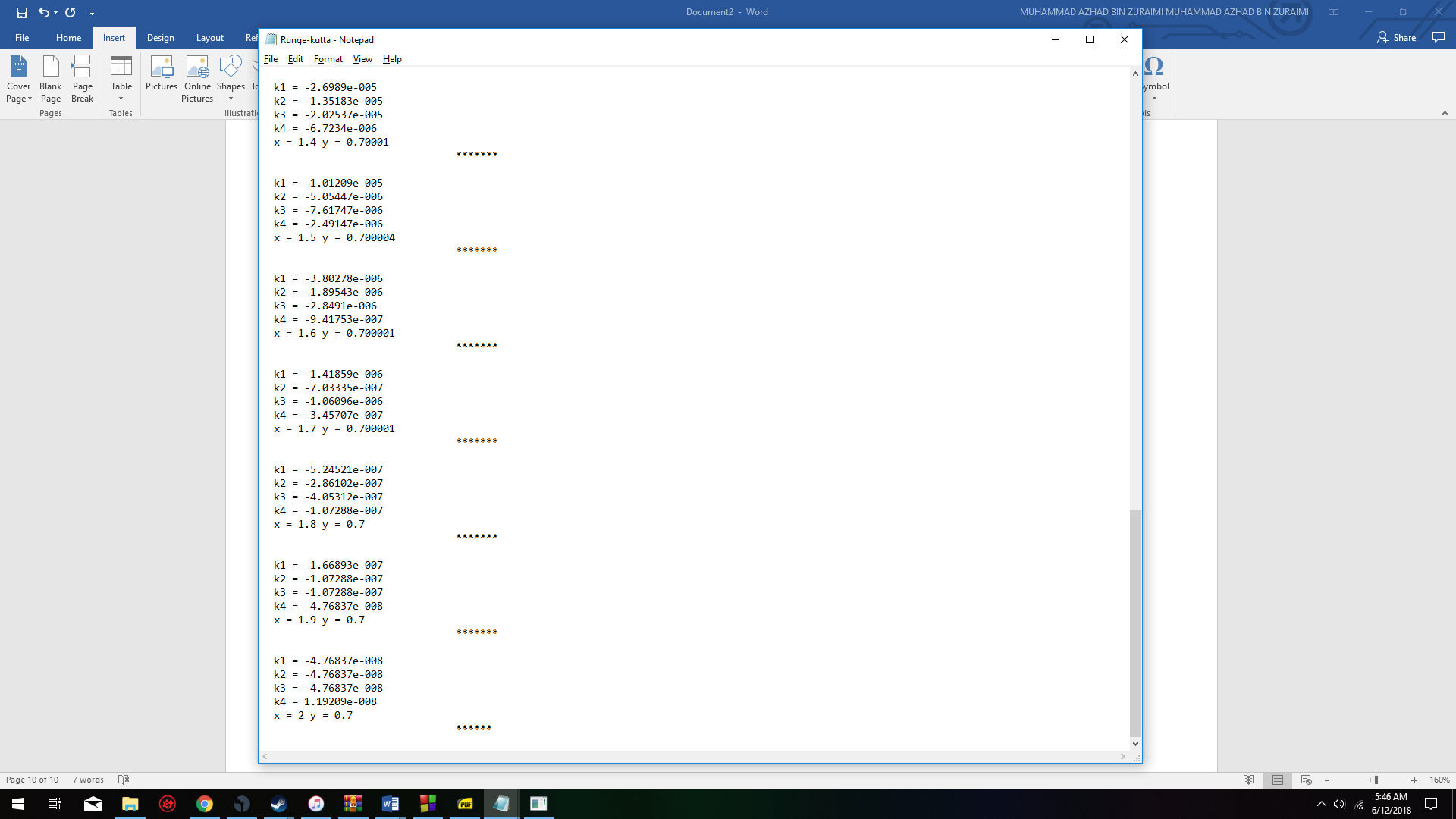




Solve RK 4







* 1. **Conclusion and Recommendation**

**Conclusion**

In the end of the mini project, students are to complete the objectives given after the program is written, executed and test run for errors. The programs will be tested on different platforms with several inputs and results. To make the finest results, students used different platform such as Flowgorithm, Microsoft Excel and others to compare the answers. With the results obtained, the first and second order differential equations can now imprint the results on the graph.

**Recommendation**

In this mini project, students recommend that always to check results before finalizing the answers. Errors occurred during the mini project, students are able to take action and corrections are made by using different platforms other than CodeBlock. Each results must be checked so that solving the First and Second Order Differential Equation could be done by using Runge-Kutta methods.