SUMMARY QUESTIONS, AND CODE SCRIPTS

1) Objective of this Project AND HOW ARE YOU GOING TO SOLVE THIS PROBLEM. CHECK THE EXAMPLES AND YOUTUBE VIDEOS BEFORE YOU MOVE TO THE STAGE OF CREATING Pseudocode. What is your basic plan to obtain the objective? (about 200 words).

From the first glance at the topic of this project, I noticed a few equations that were very familiar. I took the time to carefully read the prompt and noticed this project requires some differential equations knowledge. Luckily, I'm also taking a differential equations course at USF this semester. This means I get to test my knowledge on this concept. My first thought was how will I solve these problems mathematically on paper. Afterwards, my second thought was how will I implement this mathematical knowledge in C. With these two thoughts in mind I got to work. I first solved a few problems using my knowledge from the differential equation class. Then I proceeded to watch the YouTube video attached. Once that was done, I combined all the mathematical knowledge and started to brainstorm code ideas. This is when things got challenging… At first, imagining how I will approach this code seemed very simple, but when I started to type up the code, I started approaching many speed bumps. I continued to compute many different equations that help with solving for the unknown constants. Therefore, after a few attempts, I came up with a solid plan. I will attempt to convert what I wrote on paper, in to a detailed C script.

Please see Pseudocode & demo screenshots section below

PART 1 Script: (Used to input values of my own)

```
#include <stdio.h>

#include <math.h>


struct Root{

   double x1;

   double x2;
```

```
    double i1;
    double i2;
    int type;
};

struct TwoEquations {
    double c1;
    double c2;
};



struct Root findRoot(double a,double b,double c){
    struct Root root;
    double x1,x2,i1,i2;
    i1=0;
    i2=0;
    double delta = b*b - 4*a*c;
    if(delta<0){
        double realPart = -b/(2*a);
        double imaginaryPart = sqrt(-delta)/(2*a);
        x1 = x2 = realPart;
        i1 = i2 = imaginaryPart;
        root.type = 3;
    }
    else if(delta==0){
```

```
        x1 = x2 = -b/(2*a);
        root.type = 2;
    }
    else{
        delta = sqrt(delta);
        x1 = (-b + delta) / (2*a);
        x2 = (-b - delta) / (2*a);
        root.type = 1;
    }

    root.x1 = x1;
    root.x2 = x2;
    root.i1 = i1;
    root.i2 = i2;
    return root;
}

struct TwoEquations findTwoEquations(double a1,double b1,double c1,double a2, double b2,
double c2)  {
    struct TwoEquations result;
    float D, Dx, Dy, x, y;
    D = a1 * b2 - a2 * b1;
    Dx = c1 * b2 - c2 * b1;
    Dy = a1 * c2 - a2 * c1;
    if (D == 0) {
```

```
        if (Dx + Dy == 0)
            printf("error\n");
        else
            printf("error\n");
    }
    else {
        x = Dx / D;
        y = Dy / D;
        result.c1 = x;
        result.c2 = y;
    }
    return result;
}

int main(){

    double a1  = 6;    //input values change accordinglly
    double a2  = -9;
    double y0  = 1;
    double yi0 = 2;

    struct Root root = findRoot(1,a1,a2);

    printf("Root: \n");
    printf("x1: %.2f + %.2fi \n",  root.x1,root.i1 );
```

```
    printf("x2: %.2f + %.2fi \n",  root.x2,root.i2 );


    if(root.type == 1){         // Solving case 1: double roots
        struct TwoEquations twoEquations = findTwoEquations(1,1,y0,root.x1,root.x2,yi0);


        printf("c1: %.2f\n",twoEquations.c1);
        printf("c2: %.2f\n",twoEquations.c2);


        printf("y0(t) = %.2f(e^%.2ft) + %.2f(e^%.2ft)
\n",twoEquations.c1,root.x1,twoEquations.c2,root.x2);
    }
    else if(root.type == 2){     // Solving case 2: Repeating rootss
        double c1 = y0;
        double c2 = yi0 - c1 * root.x1;


        printf("c1: %.2f\n",c1);
        printf("c2: %.2f\n",c2);


        printf("y0(t) = (%.2f + %.2f)e^%.2f \n",c1,c2,root.x1);
    }
    else if(root.type == 3){          //Solving case 3: Complex roots


        double cCos = y0;
        double cSin = (yi0 - (cCos * root.x1)) / (- root.i1);
        double c = round(sqrt((cCos * cCos) + (cSin * cSin) ));
```

```
        double a = atan2(cSin,cCos);



        printf("cCos: %.2f \n",cCos);
        printf("cSin: %.2f \n",cSin);
        printf("c: %.2f \n",c);
        printf("a: %.2f \n",a);

        printf("y0(t) = %.2f(e^%.2f)cos(%.2f - %.2f) \n",c,root.x1,root.i1,a);

    }

return 0;

}
```

PART 2 Script: (Use text file)

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

struct Root{
    double x1;
    double x2;
    double i1;
```

```
    double i2;
    int type;
};

struct TwoEquations {
    double c1;
    double c2;
};

struct Result{
    struct Root root;
    double c1;
    double c2;

};


struct Root findRoot(double a,double b,double c){
    struct Root root;
    double x1,x2,i1,i2;
    i1 = 0;
    i2 = 0;

    double delta = b*b - 4*a*c;
    if(delta<0){
```

```
        double realPart = -b/(2*a);
        double imaginaryPart = sqrt(-delta)/(2*a);
        x1 = x2 = realPart;
        i1 = i2 = imaginaryPart;
        root.type = 3;
    }
    else if(delta==0){
        x1 = x2 = -b/(2*a);
        root.type = 2;
    }
    else{
        delta = sqrt(delta);
        x1 = (-b + delta) / (2*a);
        x2 = (-b - delta) / (2*a);
        root.type = 1;
    }

    root.x1 = x1;
    root.x2 = x2;
    root.i1 = i1;
    root.i2 = i2;
    return root;
}
```

```
struct TwoEquations findTwoEquations(double a1,double b1,double c1,double a2, double b2,
double c2)  {
    struct TwoEquations result;
    float D, Dx, Dy, x, y;
    D = a1 * b2 - a2 * b1;
    Dx = c1 * b2 - c2 * b1;
    Dy = a1 * c2 - a2 * c1;
    if (D == 0) {
        if (Dx + Dy == 0)
            printf("error\n");
        else
            printf("error\n");
    }
    else {
        x = Dx / D;
        y = Dy / D;
        result.c1 = x;
        result.c2 = y;
    }
    return result;
}


struct Result solve(double a1,double a2,double y0, double yi0){
    printf("Solving : %.2f %.2f %.2f %.2f \n",a1,a2,y0,yi0);
    struct Root root = findRoot(1,a1,a2);
```

```
    struct Result result;


    printf("Root: \n");
    printf("x1: %.2f + %.2fi \n",  root.x1,root.i1 );
    printf("x2: %.2f + %.2fi \n",  root.x2,root.i2 );


// If roots (?'s)  are nonrepeating real
    if(root.type == 1){
        struct TwoEquations twoEquations = findTwoEquations(1,1,y0,root.x1,root.x2,yi0);


        printf("c1: %.2f\n",twoEquations.c1);
        printf("c2: %.2f\n",twoEquations.c2);


        printf("y0(t) = %.2f(e^%.2ft) + %.2f(e^%.2ft)
\n",twoEquations.c1,root.x1,twoEquations.c2,root.x2);
        result.c1 = twoEquations.c1;
        result.c2 = twoEquations.c2;
        result.root = root;
    }
    // If roots (?'s)  are repeating real (?1 =?2
    else if(root.type == 2){
        double c1 = y0;
        double c2 = yi0 - c1 * root.x1;
```

```
    printf("c1: %.2f\n",c1);
    printf("c2: %.2f\n",c2);

    printf("y0(t) = (%.2f + %.2f)e^%.2f \n",c1,c2,root.x1);
    result.c1 = c1;
    result.c2 = c2;
    result.root = root;
}
// If roots (?'s)  are complex conjugate (?'s  = a+jß and a-jß
else if(root.type == 3){

    double cCos = y0;
    double cSin = (yi0 - (cCos * root.x1)) / (- root.i1);
    double c = round(sqrt((cCos * cCos) + (cSin * cSin) ));
    double a = atan2(cSin,cCos);


    printf("cCos: %.2f \n",cCos);
    printf("cCins: %.2f \n",cSin);
    printf("c: %.2f \n",c);
    printf("a: %.2f \n",a);

    printf("y0(t) = %.2f(e^%.2f)cos(%.2f - %.2f) \n",c,root.x1,root.i1,a);
    result.c1 = c;
    result.c2 = a;
```

```
        result.root = root;
    }

    return result;
}

int main(){

    FILE* inFile =  fopen("data.txt", "r");
    FILE* outFile = fopen("result.txt", "w");

    char line[256];

    // read file line by line
    while (fgets(line, sizeof(line), inFile)) {

        double a1,a2,y0,yi0;
        char *ptr;

        // split the line by ' ' to get 4 number
        // strtod : convert str to double
        char * token = strtok(line, " ");
        if(token == NULL){
            continue;
        }
```

```
        a1 = strtod(token,&ptr);
        token = strtok(NULL, " ");
        a2 = strtod(token,&ptr);
        token = strtok(NULL, " ");
        y0 = strtod(token,&ptr);
        token = strtok(NULL, " ");
        yi0 = strtod(token,&ptr);


        struct Result result = solve(a1,a2,y0,yi0);
        fprintf(outFile,"%d %.2f %.2f\n",result.root.type,result.c1,result.c2);
    }
    fclose(inFile);
    fclose(outFile);


    getchar();
    return 0;
}
```

5) List on WHAT YOU KNEW before you started this project and WHAT YOU LEARNED VIA COMPLETING THIS PROJECT. What do you think of this project? Was it difficult?

Before starting the project, I knew a few things in advance. Other than knowing many coding tricks and scripts from the previous modules, I had a decent amount of knowledge on how to solve initial value problems. This knowledge was gained from another course taken at USF better known as differential equations. After completing this project, I would like to believe I gained solid knowledge on how to compute difficult equations into C. Furthermore, I also learned how to run a text file and store values into it. Overall, this project was somewhat difficult. It seemed very simple in the beginning, but soon got very difficult.

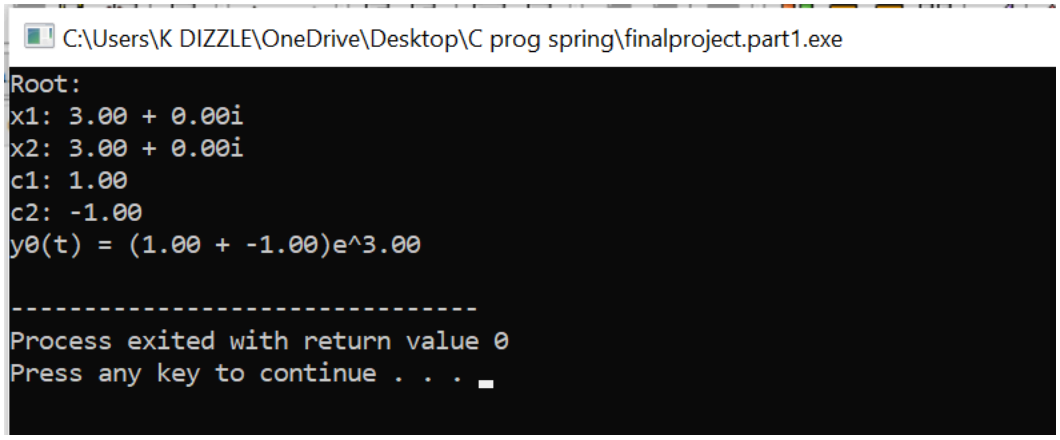
PSEUDOCODE & DEMO SCREENSHOTS

Pseudocode:

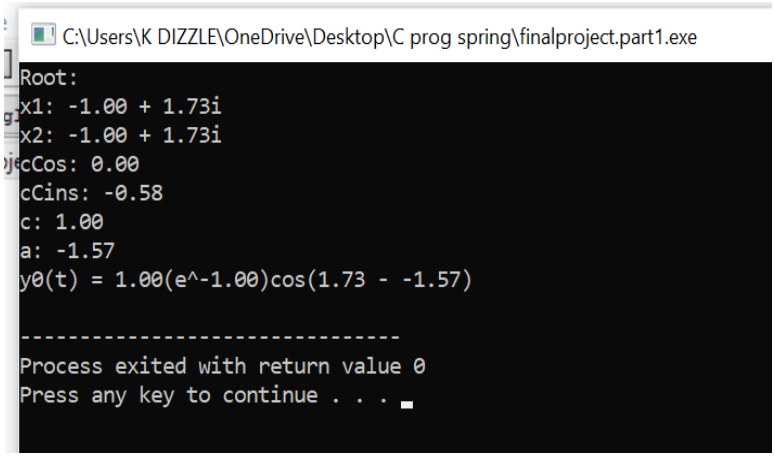


Part 1 Demo screenshots: (Values a1,a2,y0,y'0 manually picked)

a1= -6, a2= 9, y0=1, y'0= 2 :



a1= 2, a2= 4, y0=0, y'0= 1 :



Part 2 demo screenshot: (Using text file provided)

