**Deliverable 3**

CRC Model

Calculator is main class with functions for:

* Choose Function (User chooses initial function to use)
* Parse Additional Function (After choosing first and entering values, can choose to parse additional function or get answer)
* Display History (Displays calculator history of the current user since they started using calculator)
* Addition (Simple addition function)
* Subtraction (Simple subtraction function)
* Multiplication (Simple multiplication function)
* Division (Simple division function)
* History (History is an array of Functions, which are separate classes for special ones and strings for simple arithmetic)

Exponential Function is a class

* Calculate Answer (calculates value of exponential function after user enter values for a, b and x – currently x can only be an integer)
* Get Superscript (takes in string and outputs it as an exponent)
* Multiplier Number (the value of a for the function)
* Base Number (the value of b for the function)
* Exponent (the value of x for the function)
* Answer (the final answer of the function after values entered)

Pseudo Code for Exponential Function

function calculate\_answer(mult\_num, base\_num, exp\_num, answer)

answer = 1

if exp\_num is a fraction then

num = numerator of exp\_num

den = denominator of exp\_num

if num is an integer then

if den is an integer then

// denominator of fraction represents root value for base\_num

base\_num = get\_root(den, base\_num)

exp\_num = num

return calculate\_answer(mult\_num, base\_num, exp\_num, answer)

else

i = 2

tempDen = den

d = 0.0

while tempDen is not an integer

d = den

d \*= i

i += 1

tempDen = d

base\_num = get\_root(d, base\_num)

exp\_num = num \* (i- 1)

return calculate\_answer(mult\_num, base\_num, exp\_num, answer)

else

i = 2

tempNum = num

n = 0.0

while tempNum is not an integer

n = num

n \*= i

i += 1

tempNum = n

if den is an integer then

den \*= (i – 1)

base\_num = get\_root(den, base\_num)

exp\_num = n

return calculate\_answer(mult\_num, base\_num, exp\_num, answer)

else

j = 2

den \*= (i – 1)

tempDen = den

d = 0.0

while tempDen is not an integer

d = den

d \*= j

j += 1

tempDen = d

base\_num = get\_root(d, base\_num)

exp\_num = num \* (j – 1)

return calculate\_answer(mult\_num, base\_num, exp\_num, answer)

else if exp\_num is an integer then

if exp\_num >= 0 then

for i = 0 to exp\_num

answer = answer \* base\_num

answer = answer \* mult\_num

return answer

else if exp\_num < 0 then

for i = 0 to (-1 \* exp\_num)

answer = answer \* base\_num

answer = (1/answer) \* mult\_num

return answer

else

i = 2

tempExp = exp\_num

e = 0.0

while tempExp is not an integer

e = exp\_num

e \*= i

i += 1

tempExp = e

base\_num = get\_root(e, base\_num)

exp\_num \*= i - 1

return calculate\_answer(mult\_num, base\_num, exp\_num, answer)

function get\_root (root, num)

i = 1

result = False

while result == False

n = i

for index = 0 to root – 1

n \*= i

if n == num then

return i

else if n > num then

i = Root(root, num, i – 1, i)

return i

i += 1

function Root (root, num, i, j)

mid = (i + j)/2

n = mid

for index = 0 to root – 1

n \*= mid

if (n == num) OR (abs(num – n) < 0. 000000001) then

return mid

else if n > num then

return Root(root, num, i , mid)

else

return Root(root, num, mid, j)

Task0.25

Out of the following integers, choose the pair with the greatest sum of squares: 2,3,6,7,9 (Note a pair can consist of the same number, ex: 62 + 62)

ᴬᴮᶜᴰᴱᶠᴳᴴᴵᴶᴷᴸᴹᴺᴼᴾQᴿˢᵀᵁⱽᵂˣʸᶻᵃᵇᶜᵈᵉᶠᵍʰᶦʲᵏˡᵐⁿᵒᵖ۹ʳˢᵗᵘᵛʷˣʸᶻ⁰¹²³⁴⁵⁶⁷⁸⁹⁺⁻⁼⁽⁾./