Linked List class:

class List

{

public:

List\* head;

void addList(List\* head);

void deleteList(List\* head);

List() = default;

~List();

}

Node class:

class Node

{

public:

Node() = default;

~Node();

private:

int coefficient;

int exponent;

Node\* ptr;

{

Function: deriveFunct

Return: none

Parameter: list

Logic:

* Create pointer to beginning of linked list
* Traverse using pointer
* if(ptr -> next)
  + deriveFunct(ptr->next)
* Store coefficient
* Look for exponent
  + Subtract 1 from exponent
  + Store exponent \* coefficient into coefficient
* if sin or cos or tan
  + differentiate accordingly (i.e. derivative of sin is cos)
* printDerivative

Function: printDerivative

Returns: none

Parameters: list

Logic:

* Connect to derive.txt
* Create pointer to beginning to linked list
* If(ptr->next)
  + printDerivative(ptr->next)
* while ptr->next
  + print contents of each node to file
* print ‘\n’ to file

Function: main

Returns: none

Parameters: none

Logic:

* Create pointers to classes
  + List\* ptrList;
  + Node\* ptrNode;
* Connect to input file functions.txt
* while not EOF
  + read from functions.txt
    - if num of operators == 0
      * count how many operators in current line
    - if num operators < num of nodes
      * counter = 0
      * while counter > num nodes – num operators
        + List.deleteList
    - if num operators > num of nodes
      * counter = 0;
      * while counter < num operators – num nodes
        + List.addList
    - for < num of operators
      * create node
      * add to linked list
    - deriveFunct