1. **A text document containing an explanation of your designed data type, your specification and your pseudo-code for all operations.**

**Explanation of data type:**

typedef struct

{

polyNode\* head;

polyNode\* current;

} polynomial;

In our data type, named “polynomial”, there are two nodes named “head” and ”current” these are of type “polyNode”

typedef struct n {

term d; // STORE A TERM IN THE NODE

struct n\* successor;

} polyNode;

The data type “polyNode” contains two things, the first is of data type “term” named “d”.

The second is creating a pointer to an empty node.

The data type “term” contains a double and an int, named “coefficient” and “exponent”

**Your specification:**

**Pseudo-code for all operations**

**Pseudo-code for mainProgram.c:**

Create two polynomial pointers named “poly1” and “poly2” and set both values to null.

Print to screen that the program is creating new polynomials and to input the values for the first and second polynomial.

Set “poly1” and “poly2” equal to the return of the method createPoly after each time it’s called.

Display “poly1” and “poly2” that were just created on the screen using the displayPoly function and passing in the “poly1” and “poly2” variable names that was just created.

Print to the screen that the program is Adding the polynomials.

Call the addPoly function and pass in both “poly1” and “poly2”

Print to the screen that the program is Subtracting the polynomials.

Call the subPoly function and pass in both “poly1” and “poly2”

Print to the screen that the program is Multiplying the polynomial.

Call the multiplyPoly function and pass in “poly1”

Print to the screen that the program is Dividing the polynomial.

Call the dividePoly function and pass in “poly1”

Print to the screen that the program is normalizing the polynomial.

//NOT SURE YET

Print to the screen that the program is returning the order of polynomial

Call the orderPoly function and pass in the “poly1” variable.

Print to the screen that the program is deleting polynomial.

Call the deletePoly function and pass in the “poly1” variable.

Print to screen that the program is displaying the polynomial

Call the displayPoly function and pass in the “poly1” variable.

Print to the screen that we are exiting the program

Return exit success to indicate successful program execution status.

**Pseudo-code for polynomial.h:**

#indef \*\*\*\*\*\*\*\*\*\*\*cant remember what these are called

#define \*\*\*\*\*\*\*\*\*\* fill in pls

Create an enum called polyError that contains error codes for the linked list, It contains ok, illegalNode, and noMemory

Create a struct called term that contains a double and an int called coefficient and exponent

Create a struct for a node called polyNode that contains a term and a struct with pointer to n called d and successor

Create a struct called polynomial that contains a polyNode pointer called head and a polyNode pointer called current

Declare the createPoly function that returns a polynomial pointer

Declare deletePoly which deletes the polynomial that’s passed in

Declare the addPoly function that returns a polynomial pointer after it does the addition on two polynomials that are passed in

Declare the dubtractPoly function that returns a polynomial pointer after it does the subtraction on two polynomials that are passed in

Declare the multiplyPoly function that returns a polynomial pointer after it does the multiplication of the polynomial passed in

Declare the dividePoly function that returns a polynomial pointer after it does the division of the polynomial passed in

Declare the normalisePoly function that returns a polynomial pointer after it normalizes the polynomial that is passed in

Declare the function orderPoly that orders the polynomial that is passed in

Declare the function displayPoly that displays the polynomial that is passed in

Declare the accessData function that returns a term pointer for the polynomial that is passed in

Declare the insertAfter function that returns a polyError after the polynomial and an int exponent is passed in

Declare the gotoHead which sets the pointer to the head node of the linked list for the polynomial that is passed in

Declare the gotoNextNode which returns a polyError for the polynomial that is passed in

#endif

**Pseudo-code for the makefile:**

List all the targets that need to be executed in the makefile inside the exec

Set variables for use in the makefile, one called poly used for polynomial and one called main used for mainProgram

Set the variable name all to the elements of the exec

Compile the polynomial.c and the polynomial.h file into polynomial.o using gcc -Wall -ggdb -

Compile the mainProgram.c file into mainProgram.o using gcc -Wall -ggdb -

Link the components to create the project executable, do this by using the two object files that were just created using gcc -o project

Run the application

Remove all object files and the final project file to set a clean target, using rm -f