3740 group project By Kegan Rieder and Justin Wolfenden

We dealt with this issue by converting the polynomial to the representation that is easiest for the current task, we would then perform the functions using this representation and if necessary, we would change it back to the original representation before returning the value. You can see this demonstrated through add, when it encounters multiple sparse, it will convert these sparse to dense, it will then take the answer it got from the calculation and convert that to sparse before returning it to the user.

A brief explanation of the implementation of all functions

**Is-sparse?(Polynomial)**

Polynomial is the polynomial you want to see if it is sparse.

It works by first checking if the list representation of the polynomial is an empty list. If so it returns false, meaning its not sparse.

Next it checks if the first item in the list is also a list, if it is it will return true (the list is sparse), if the first item in the list is not a list it returns false (the list is not sparse).

**Make-sparse(polynomial index)**

Polynomial is the list representation of the polynomial that you want to make sparse (list given must be dense). Index should be set to 0 and is used for recursion

It works by first checking if the list is empty, if it is it will just return ()

If the list is not empty it will then check if the next item in the list is 0, if it is 0 it will not add it to the new sparse list, it will then run make\_sparse((cdr polynomial) (add1 index)) in order to check the next item in the list with the corresponding index

if the next item in the list is not 0 it will add a small list consisting of the current number and the index or coeficent of that number to the new list and use recursion make\_sparse((cdr polynomial) (add1 index)) in order to check the next item in the list with the corresponding index

after the list is empty it will return the newly created list.

**To-sparse(Polynomial, index)**

Polynomial is the polynomial you want to change to sparse index is set to 0 for recursion

It will first use is\_sparse?(polynomial) to check if the polynomial is already sparse If it is sparse it will return the polynomial given to it, if it is not it will run make\_sparse(polynomial index) to make it sparse

**Is-dense?(x)**

x is the polynomial you want to see if it is dense.

checks if the first item in the list is also a list, if it is it will return false (the list is not dense), if the first item in the list is not a list it returns true (the list is dense).

**Make-dense(x y)**

X is the polynomial you want to make dense (list given must be sparse), y is set to 0 and used for recursion

First it will check if the list is empty, if it is it will return null

If the list is not empty it will check if y is equal to the second item present int the first list of the sparse list, if they are equal then it will add the first item of the first list to the new list , it will then run make\_dense((cdr x) (+ 1 y)) in order to check the next item in the list with the corresponding y.

If y is not equal to the second item of the first list in sparse then it will add 0 to the new list, it will then run make\_dense(( x) (+ 1 y)) in order to keep y updated to the correct index.

**To-dense(x, y)**

x is the polynomial you want to change to sparse y is set to 0 for recursion

It will first use is\_dense?(x) to check if the polynomial is already sparse If it is dense it will return the polynomial given to it, if it is not it will run make\_dense(x y) to make it dense