Name: Saurabh Mukherjee Roll no: 001910501006 Class: BCSE II Sem: First Session 2020-21 Assignment Set :2

Problem No. 1

Problem Statement :

Define an ADT for Polynomials. Write C data structure representation and functions for the operations on the Polynomials in a Header file. Write a menu-driven main program in a separate file for testing the different operations and include the above header file.

Solution Approach:

An array of structs which has two date fields one for the coefficient and other for the exponent is taken as the base structure of the polynomial.We keep the polynomials in linear fashion that is , once the first polynomial ends we keep the next polynomial right from the adjacent location to the first poly in the array. Then we are performing the operations such as addition ,subtraction and multiplication and storing the final result into the same array right from the index where the second poly ends .One thing to note is that we must enter the polynomial in sequential order say ascending order so that we can have an efficient implementation of the algorithm for multiplication and addition.

Structured Pseudocode :

1.Define a polynomial structure with exponent and coefficient as the values.

2.Create an array of poly type say poly[MAX].

3.Initialise a=number of terms in first poly

4.Initialise b=number of terms in second poly

5.Call Mutiply(poly \*p[MAX],a ,b)

6.For Add call add(poly \*p[MAX],a,b)

7.for subtract call subtract(poly \*p[MAX],a,b)

Mutiply(poly \*p[MAX],a ,b)

1.for i from 0 to a

2. for j from a to b

3. result= (\*p)[i] \* (\*p)[j] .

4. check if the exponent matches with any previously multiplied terms

5. if exponent matches

6. update that coeff to coeff+result

7. else

8. store the result in a new index of the poly array

add(poly \*p[MAX],a,b)

1.Initialise i=0,j=0

2.while i in range a and j from a to b

3. if exp ((\*p)[i] )<exp((\*p)[j]) , copy (\*p)[i] in poly and increment i

4. else if exp ((\*p)[i] )>exp((\*p)[j]), copy (\*p)[i] in poly and increment j

5. else add the coeff of (\*p)[i] and (\*p)[i] and store in poly,increment i and j

For subtract we need to follow the same pseudocode as addition ,with only difference in the sign ,positive being replaced by negative

Results:

We get the result to be stored in the same poly array as the initial with the index values being modified and then later accessing the array of poly to retrieve the date we just need to loop through it once.

Discussion:

We must take the polynomial array with increasing exponent say 0(constant term first if present ,then the other terms ).We need to pass the address of the poly to the functions such as multiply ,add or subtract in order to get the changes in the final array once the scope of the function block is over.

We preinitialise the size of the poly array with a macro say MAX and it must be ensured that we would be able to store all the things such as the first poly , the second polynomial and the result all in the same array of poly;

Separate files containing commented source code

The file has been attached.