Koneru Lakshmaiah Education Foundation

(Deemed to be University)

FRESHMAN ENGINEERING DEPARTMENT

A Project Based Lab Report

On

Implementation of Linked List Applications

SUBMITTED BY:

I.D NO NAME

2200080183 | PATTAN MAHAMMAD SALMAN KHAN

2200080184 | MUNAGALA PAVAN TEJA

2200080185 | SUGGU SRI HARSHA

2200080186 | VEGI BALA BHARGAVI

UNDER THE GUIDANCE OF

Dr. Bhaskar Marapelli

Associate Professor

BES - 1

KL UNIVERSITY

Green fields, Vaddeswaram – 522 302

Guntur Dt., AP, India.

DEPARTMENT OF BASIC ENGINEERING SCIENCES-1

CERTIFICATE

This is to certify that the project based laboratory report entitled

"**Implementation of Linked List Applications**"

submitted by

Mr. PATTAN MAHAMMAD SALMAN KHAN of bearing Regd. No. 2200080183,

Mr. MUNAGALA PAVAN TEJA of bearing Regd. No. 2200080184,

Mr. SUGGU SRI HARSHA of bearing Regd. No. 2200080185,

Ms. VEGI BALA BHARGAVI of bearing Regd. No. 2200080186,

to the **Department of Basic Engineering Sciences-1, KL University** in partial fulfillment of the requirements for the completion of a project-based Laboratory in "COMPUTATIONAL THINKING FOR STRUCTURED DESIGN" course in I B Tech I Semester, is a bonafide record of the work carried out by him/her under my supervision during the academic year 2022 – 2023.

PROJECT SUPERVISOR	HEAD OF THE DEPARTMENT
Dr. Bhaskar Marapelli	Dr. D. Haritha

ACKNOWLEDGEMENTS

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project-based laboratory report.

I express the sincere gratitude to our principal **Dr. A. Jagadeesh** for his administration towards our academic growth.

I express sincere gratitude to HOD-BES-1 **Dr. D. Haritha** for her leadership and constant motivation provided in successful completion of our academic semester. I record it as my privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to our project supervisor Dr.Bhaskar Marapelli for his/her novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

Name: P.M. SALMAN KHAN Regd.NO: 2200080183

ABSTRACT

The main aim of this project is to implement one of the most important applications of linked list such as polynomial operations such as addition, subtraction, multiplication and derivation. To perform these operations each polynomial needs to represent in one linked list and each node in the list contains three parts to store coefficient, exponent and link to next term of polynomial respectively.

There are four modules in in this project:

- polynomial Addition
- polynomial subtraction
- polynomial multiplication
- polynomial derivation

This project is a patch of Pointer, Self-Referential Structures, Linked List. Using this it is possible to perform Polynomial Addition, Subtraction, Multiplication and Derivation. This is only can perform 1-Variable Polynomial Linked List Applications.

INDEX

INTRODUCTION

Polynomial can be represented in the various ways. These are:

- Using arrays
- Using Linked List

But here we are going implement this application using Linked List Data Structure with Self-Referential Data Structure (struct).

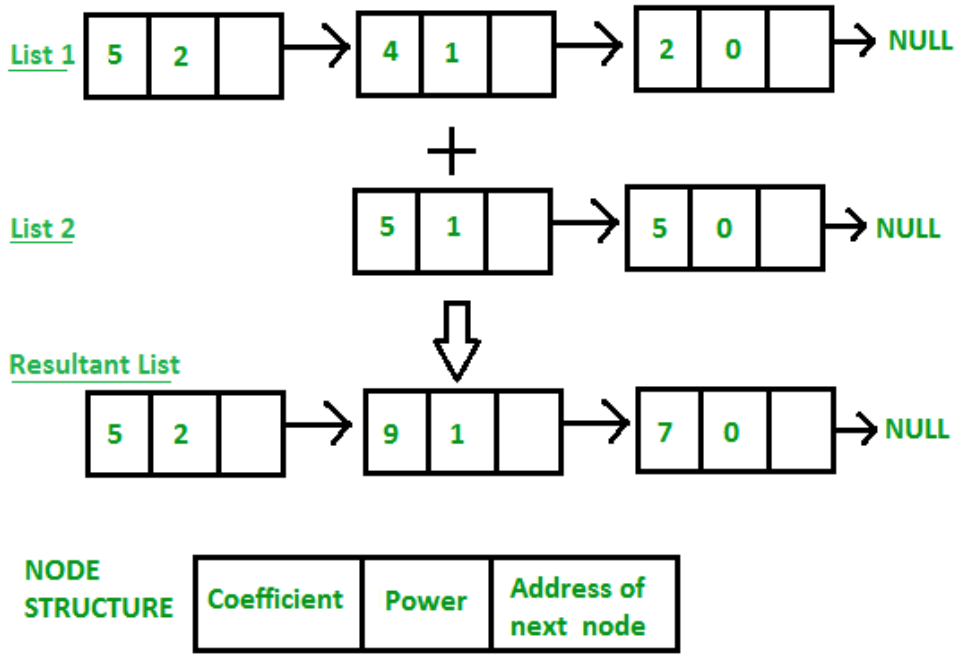
Representation using linked list:

- We can represent a polynomial using a linked list where the nodes contain 2 data fields (Coefficient, Exponent) and one reference to the next node.

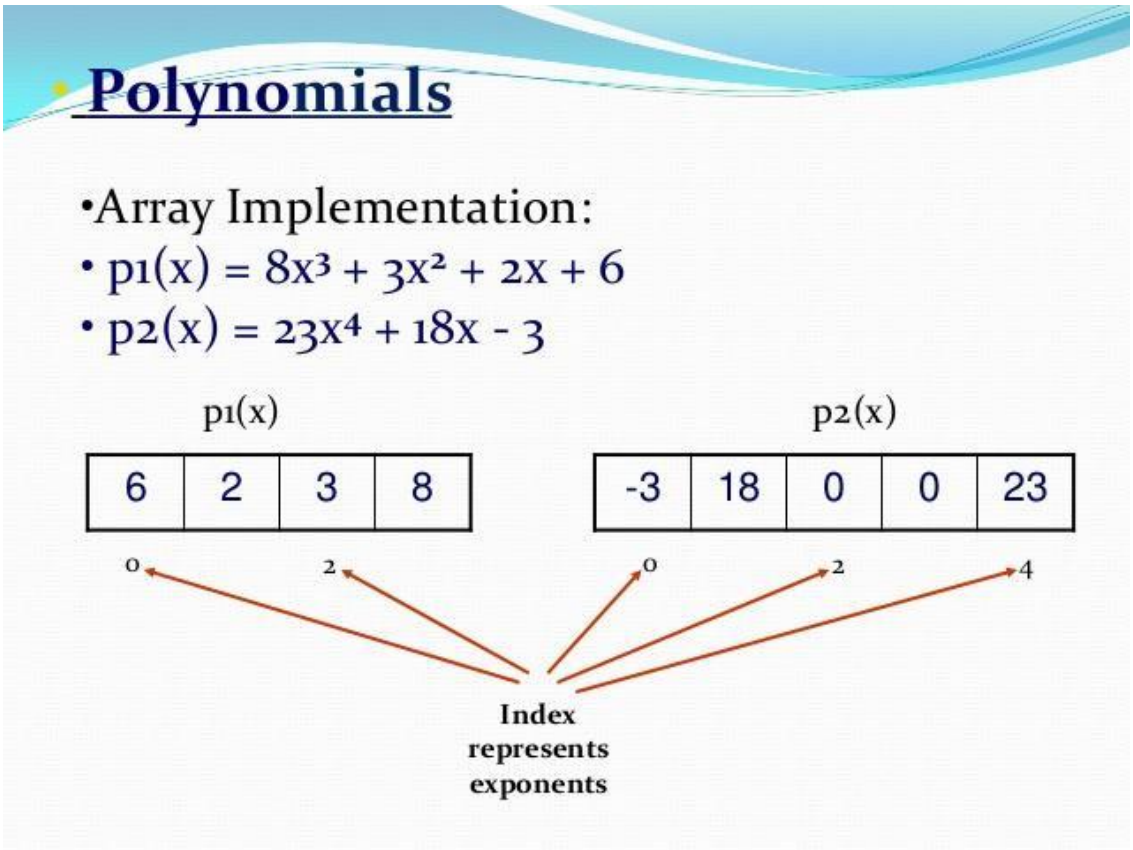
Methods: -
Example Case: Polynomial Addition

- This method of implementation uses a dynamic memory allocation. Memory is allocated only when needed hence, the memory required is exactly equal to the amount of memory needed.
- Thus, memory is used more efficiently. Also, while adding the two polynomials adding a term as node at the end of a linked list is more convenient then adding the information of the node in four separate arrays.

Linked List representation of a polynomial (for 1 variable):



Array representation of a polynomial (for 1 variable):



AIM

The main aim of this project is to implement one of the most important applications of linked list such as polynomial operations such as addition, subtraction, multiplication and derivation. To perform these operations each polynomial needs to represent in one linked list and each node in the list contains three parts to store coefficient, exponent and link to next term of polynomial respectively. There are four modules in in this project

- polynomial Addition
- polynomial subtraction
- polynomial multiplication
- polynomial derivation

Using

- creating using linked lists
- pointers, self-referential structures

Advantages: -

- Easier to Perform:
    1. Polynomial Addition
    2. Polynomial Subtraction
    3. Polynomial Multiplication
    4. Polynomial Derivation

Disadvantages: -

- Not designed to perform more than 2 polynomials
- Not designed for multiple variables

Future enhancements: -

- Addition of performing more than 2 Polynomials.
- Avoiding 0 Values printing in the results.
- Optimizing the structure of the code.
- Running inside a loop.
- Implementation of Graphical User Interface (GUI).

SYSTEM REQUIREMENTS

➢ SOFTWARE REQUIREMENTS:

The major software requirements of the project are as follows:

Language: C

IDE: Visual Studio Code

Operating system**:** Windows, Linux, MacOS with Terminal Access and Gcc.

➢ HARDWARE REQUIREMENTS:

The hardware requirements that map towards the software are as follows:

RAM: 2GB Ram Minimum

Processor: Intel core duo, Pentium Minimum

ALGORITHM

Step 1: Start

Step 2: Printing Available Options.

Step 3: Taking response and storing it in an integer data type variable.

Step 4: Using Switch Case to Run the Preferred Program.

Step 5: if the case not in the range, Prints Invalid argument

Step 6: Stop

Example:

PolyAdd Function:

Step 1: Function call with parameters of Create (head1) and Create(head2).

Step 2: Function call to take input and insert into the nodes using insert (head, co, exp) for both head1 and head2.

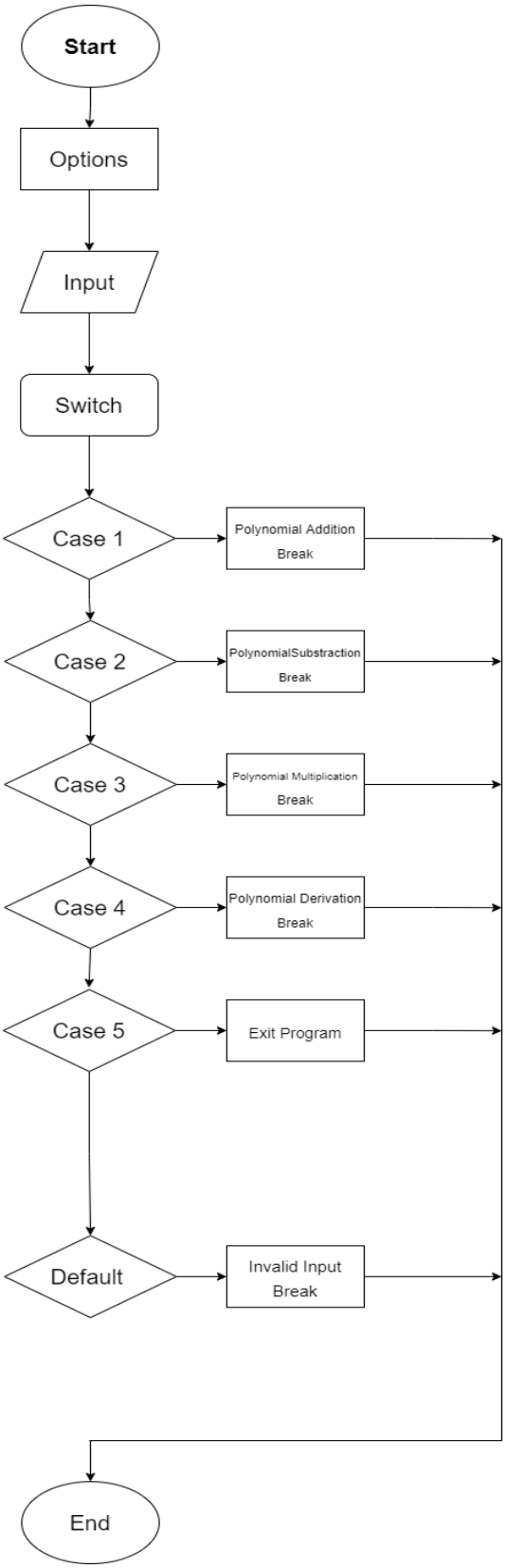
Step 3: PolyAdd Function call in this case.

Step 4: printing the result using print function.

Step 5: Break

Same Process is repeat for the rest of the application, but in the case polynomial derivation, only head1 will be used store and the inputs.

FLOW CHART


Start
Options
Input
Switch
Case 1
Polynomial Addition
Break
Case 2
PolynomialSubstraction
Break
Case 3
Polynomial Multiplication
Break
Case 4
Polynomial Derivation
Break
Case 5
Exit Program
Default
Invalid Input
Break
End

IMPLEMENTATION

```
#include <stdio.h>
#include <stdlib.h>

// structure define
struct node {
   float coeff;
   int expo;
   struct node *link;
};

// insert function inserts all the data scanned from create function
struct node* insert(struct node* head, float co, int ex){
   struct node* temp;
   struct node* newP = malloc(sizeof(struct node));
   newP -> coeff = co;
   newP -> expo = ex;
   newP -> link = NULL;

   if (head == NULL || ex > head -> expo){
      newP -> link = head;
      head = newP;
   }
   else{
      temp = head;
      while (temp -> link != NULL && temp -> link -> expo >= ex){
         temp = temp -> link;
      }
      newP -> link = temp -> link;
      temp -> link = newP;
   }
   return head;
}
```

```c
// create function creates nodes
struct node* create (struct node* head){
   int i, n, expo;
   float coeff;

   printf("Enter the number of terms: ");
   scanf("%d", &n);

   for (i = 0; i < n; i++){
      printf("Enter the coefficient for term %d : ", i + 1);
      scanf("%f", &coeff);

      printf("Enter the exponent for term %d : ", i + 1);
      scanf("%d", &expo);

      head = insert(head, coeff, expo);
   }
   return head;
}

// print function prints the polynomial with conditions
void print(struct node* head) {
   if (head == NULL){
      printf("Empty polynomial");
   }
   else {
      struct node* temp = head;
      while (temp != NULL){
         printf("%.1fx^%d", temp -> coeff, temp -> expo);
         temp = temp -> link;
         if (temp != NULL){
            if (temp -> coeff > 0){
               printf(" + ");
            }
            else {
               printf(" ");
            }
         }
         else {
            printf("\n");
         }
      }
   }
}
```

```
// Polynomial Addition
void polyAdd (struct node* head1, struct node* head2){
   struct node* ptr1 = head1;
   struct node* ptr2 = head2;
   struct node* head3 = NULL;

   while (ptr1 != NULL && ptr2 != NULL){
      if (ptr1 -> expo == ptr2 -> expo){
         head3 = insert(head3, ptr1 -> coeff + ptr2 -> coeff, ptr1 -> expo);
         ptr1 = ptr1 -> link;
         ptr2 = ptr2 -> link;
      }
      else if (ptr1 -> expo > ptr2 -> expo){
         head3 = insert(head3, ptr1 -> coeff, ptr1 -> expo);
         ptr1 = ptr1 -> link;
      }
      else if (ptr1 -> expo < ptr2 -> expo) {
         head3 = insert(head3, ptr2 -> coeff, ptr2 -> expo);
         ptr2 = ptr2 -> link;
      }
   }

   while (ptr1 != NULL){
      head3 = insert(head3, ptr1 -> coeff, ptr1 -> expo);
      ptr1 = ptr1 -> link;
   }

   while (ptr2 != NULL){
      head3 = insert(head3, ptr2 -> coeff, ptr2 -> expo);
      ptr2 = ptr2 -> link;
   }

   printf("The first polynomial is:\n\t\t ");
   print(head1);
   printf("\n");
   printf("The second polynomial is:\n\t\t ");
   print(head2);
   printf("\n");
   printf("The sum of the two polynomials is:\n\t\t");
   print(head3);
   printf("\n");
}
```

```
// Polynomial Subtraction
void polySub (struct node* head1, struct node* head2){
   struct node* ptr1 = head1;
   struct node* ptr2 = head2;
   struct node* head3 = NULL;

   while (ptr1 != NULL && ptr2 != NULL){
      if (ptr1 -> expo == ptr2 -> expo){
         head3 = insert(head3, ptr1 -> coeff - ptr2 -> coeff, ptr1 -> expo);
         ptr1 = ptr1 -> link;
         ptr2 = ptr2 -> link;
      }
      else if (ptr1 -> expo > ptr2 -> expo){
         head3 = insert(head3, ptr1 -> coeff, ptr1 -> expo);
         ptr1 = ptr1 -> link;
      }
      else if (ptr1 -> expo < ptr2 -> expo) {
         head3 = insert(head3, -ptr2 -> coeff, ptr2 -> expo);
         ptr2 = ptr2 -> link;
      }
   }

   while (ptr1 != NULL){
      head3 = insert(head3, ptr1 -> coeff, ptr1 -> expo);
      ptr1 = ptr1 -> link;
   }

   while (ptr2 != NULL){
      head3 = insert(head3, ptr2 -> coeff, ptr2 -> expo);
      ptr2 = ptr2 -> link;
   }

   printf("The first polynomial is: ");
   print(head1);
   printf("\n");
   printf("The second polynomial is: ");
   print(head2);
   printf("\n");
   printf("The subtraction of the two polynomials is: ");
   print(head3);
   printf("\n");
}




// Polynomial Multiplication
```

```
void polyMult (struct node* head1, struct node* head2){
   struct node* ptr1 = head1;
   struct node* ptr2 = head2;
   struct node* head3 = NULL;

   if(head1 == NULL || head2 == NULL){
      printf("Zero Polynomials");
      return ;
   }

   while (ptr1 != NULL){
      while (ptr2 != NULL){
         head3 = insert(head3, ptr1 -> coeff * ptr2 -> coeff, ptr1 -> expo + ptr2 -> expo);
         ptr2 = ptr2 -> link;
      }
      ptr1 = ptr1 -> link;
      ptr2 = head2;
   }

   struct node* ptr3 = head3;
   struct node* temp = NULL;

   while (ptr3 -> link != NULL) {
      if (ptr3 -> expo == ptr3 -> link -> expo){
         ptr3 -> coeff = ptr3 -> coeff + ptr3 -> link -> coeff;
         temp = ptr3 -> link;
         ptr3 -> link = ptr3 -> link -> link;
         free(temp);
      }
      else {
         ptr3 = ptr3 -> link;
      }
   }
   printf("The first polynomial is: ");
   print(head1);
   printf("\n");
   printf("The second polynomial is: ");
   print(head2);
   printf("\n");
   printf("The multiplication of the two polynomials is: ");
   print(head3);

}



// Polynomial Derivation
void polyDeriv (struct node* head1){
```

```
   struct node* ptr1 = head1;
   struct node* head3 = NULL;

   while (ptr1 != NULL){
      head3 = insert(head3, ptr1 -> coeff * ptr1 -> expo, ptr1 -> expo - 1);
      ptr1 = ptr1 -> link;
   }
   printf("The first polynomial is: ");
   print(head1);
   printf("\n");
   printf("The derivative of the polynomial is: ");
   print(head3);
   printf("\n");
}


// Root
int main(){
   struct node* head1 = NULL;
   struct node* head2 = NULL;

   int op;

   printf("\nOperations Available:\n1 - Polynomial Addition\n2 - Polynomial Subtraction\n3 - Polynomial
Multiplication\n4 - Polynomial Derivation\n5 - To exit the program\n");
   printf("Enter the operation you want to perform: ");
   scanf("%d", &op);

   switch (op) {
   case 1 :
      printf("Enter the first polynomial: \n");
      printf("\n");
      head1 = create(head1);
      printf("\n");
      printf("Enter the second polynomial: \n");
      printf("\n");
      head2 = create(head2);
      printf("\n");
      polyAdd(head1, head2);
      printf("\n");
      break;

   case 2 :
      printf("Enter the first polynomial: \n");
      printf("\n");
      head1 = create(head1);
      printf("\n");
      printf("Enter the second polynomial: \n");
```

```
      printf("\n");
      head2 = create(head2);
      printf("\n");
      polySub(head1, head2);
      printf("\n");
      break;

    case 3 :
      printf("Enter the first polynomial: \n");
      printf("\n");
      head1 = create(head1);
      printf("\n");
      printf("Enter the second polynomial: \n");
      printf("\n");
      head2 = create(head2);
      printf("\n");
      polyMult(head1, head2);
      printf("\n");
      break;

    case 4 :
      printf("Enter the polynomial: \n");
      printf("\n");
      head1 = create(head1);
      printf("\n");
      polyDeriv(head1);
      printf("\n");
      break;

    case 5 :
      printf("Exiting the program...");
      break;

    default:
      printf("Invalid Input, Please Retry Again!");
      break;
    }

    return 0;
}
```

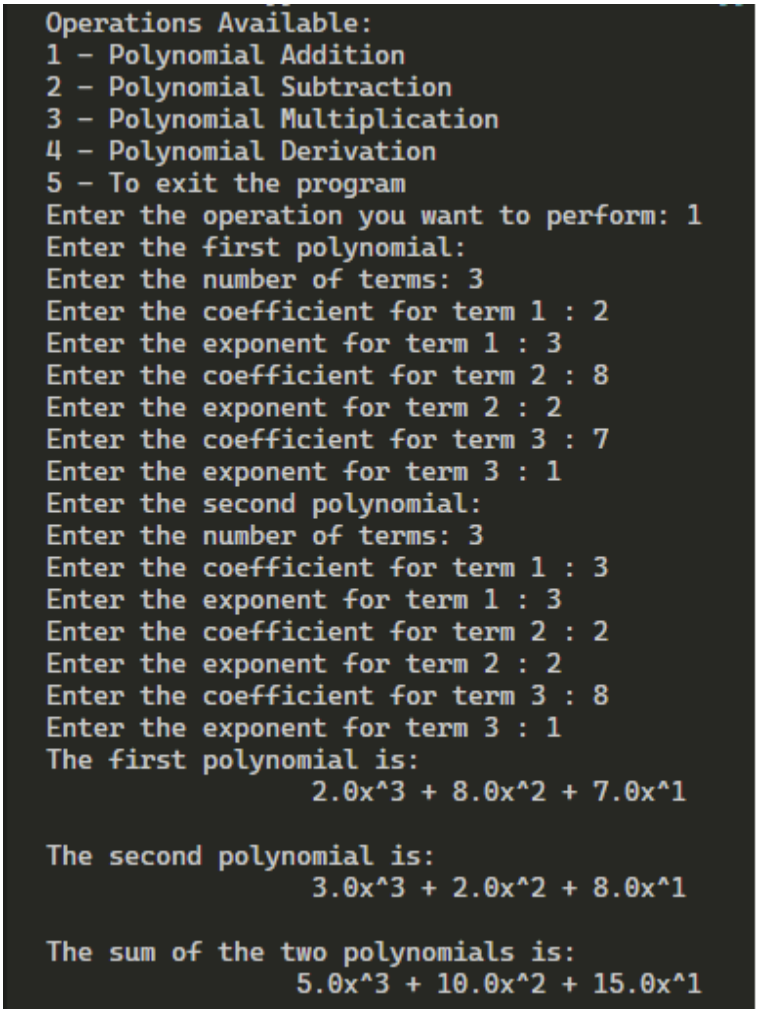
//End of the Program

RESULTS AND SCREENSHOTS

OUTPUTS

Screen Shots:

1. Polynomial Addition



2. Polynomial Substraction

```
Operations Available:
1 - Polynomial Addition
2 - Polynomial Subtraction
3 - Polynomial Multiplication
4 - Polynomial Derivation
5 - To exit the program
Enter the operation you want to perform: 2
Enter the first polynomial:
Enter the number of terms: 3
Enter the coefficient for term 1 : 2
Enter the exponent for term 1 : 3
Enter the coefficient for term 2 : 6
Enter the exponent for term 2 : 2
Enter the coefficient for term 3 : 5
Enter the exponent for term 3 : 1
Enter the second polynomial:
Enter the number of terms: 3
Enter the coefficient for term 1 : 2
Enter the exponent for term 1 : 3
Enter the coefficient for term 2 : 2
Enter the exponent for term 2 : 2
Enter the coefficient for term 3 : 9
Enter the exponent for term 3 : 1
The first polynomial is: 2.0x^3 + 6.0x^2 + 5.0x^1

The second polynomial is: 2.0x^3 + 2.0x^2 + 9.0x^1

The subtraction of the two polynomials is: 0.0x^3 + 4.0x^2 -4.0x^1
```

3. Polynomial Multiplication

```
Operations Available:
1 - Polynomial Addition
2 - Polynomial Subtraction
3 - Polynomial Multiplication
4 - Polynomial Derivation
5 - To exit the program
Enter the operation you want to perform: 3
Enter the first polynomial:
Enter the number of terms: 3
Enter the coefficient for term 1 : 5
Enter the exponent for term 1 : 3
Enter the coefficient for term 2 : 2
Enter the exponent for term 2 : 2
Enter the coefficient for term 3 : 6
Enter the exponent for term 3 : 1
Enter the second polynomial:
Enter the number of terms: 3
Enter the coefficient for term 1 : 5
Enter the exponent for term 1 :
3
Enter the coefficient for term 2 : 5
Enter the exponent for term 2 : 2
Enter the coefficient for term 3 : 4
Enter the exponent for term 3 : 1
The first polynomial is: 5.0x^3 + 2.0x^2 + 6.0x^1

The second polynomial is: 5.0x^3 + 5.0x^2 + 4.0x^1

The multiplication of the two polynomials is: 25.0x^6 + 35.0x^5 + 60.0x^4 + 38.0x^3 + 24.0x^2
```

4. Polynomial Derivation

```
Operations Available:
1 - Polynomial Addition
2 - Polynomial Subtraction
3 - Polynomial Multiplication
4 - Polynomial Derivation
5 - To exit the program
Enter the operation you want to perform: 4
Enter the polynomial:
Enter the number of terms: 3
Enter the coefficient for term 1 : 5
Enter the exponent for term 1 : 3
Enter the coefficient for term 2 : 4
Enter the exponent for term 2 : 2
Enter the coefficient for term 3 : 1
Enter the exponent for term 3 : 1
The first polynomial is: 5.0x^3 + 4.0x^2 + 1.0x^1

The derivative of the polynomial is: 15.0x^2 + 8.0x^1 + 1.0x^0
```

5. Exiting The Program

```
Operations Available:
1 - Polynomial Addition
2 - Polynomial Subtraction
3 - Polynomial Multiplication
4 - Polynomial Derivation
5 - To exit the program
Enter the operation you want to perform: 5
Exiting the program...
```

Default

```
Operations Available:
1 - Polynomial Addition
2 - Polynomial Subtraction
3 - Polynomial Multiplication
4 - Polynomial Derivation
5 - To exit the program
Enter the operation you want to perform: 8
Invalid Input, Please Retry Again!
```

CONCLUSION

In the process of making this program successful. We got to learn linked list data structure, pointers, self-referential structure (struct) and head, heap, memory allocation.

It was hard at first, but under the guidance of Dr. M. Bhaskar, It became easy for us to understand and implement. Before this Project we had a basic knowledge of Number arithmetic's.

But after this project we are aware of Polynomial Arithmetic's implementations in Programming Languages.

Using this Program,

We are successfully able to implement and perform

- Polynomial Addition
- Polynomial Substruction
- Polynomial Multiplication
- Polynomial Derivation

We promise this project would not become a legacy, as the knowledge we gain we will work, optimize it.

Thank You all for giving this Wonderful Opportunity for helping us to test our knowledge in the Real-World Cases.