2. APPLICATION OF LIST ADT (POLYNOMIAL ADDITION)

Preamble

The list data structures are widely used in:

Sorting Algorithms: Lists are essential to constructing efficient sorting algorithms such as quick sort and merge sort.

Data Analytics: Lists are often used to represent datasets in data analytics and machine learning.

Database Management Systems

Steps

Step 1: Define structure variables and functions.

Step 2: Get the values for two polynomial expression with co efficient and term.

Step 3: Add two terms when the co efficient are same. If not store the higher order in the resultant polynomial.

Step 4: Display the result of addition.

Implementation in C

```
/* declare three arrays p1, p2, p3 of type structure poly.
* each polynomial can have maximum of ten terms
* addition result of p1 and p2 is stored in p3 */
struct poly p1[10],p2[10],p3[10];
/* function prototypes */
int readPoly(struct poly []);
int addPoly(struct poly [],struct poly [],int ,int ,struct poly []);
void displayPoly( struct poly [],int terms);
int main()
{
      int t1,t2,t3;
      /* read and display first polynomial */
      t1=readPoly(p1);
      printf(" \n First polynomial : ");
```

```
	displayPoly(p1,t1);
	/* read and display second polynomial */
	t2=readPoly(p2);
	printf(" \n Second polynomial : ");
	displayPoly(p2,t2);
	/* add two polynomials and display resultant polynomial */
	t3=addPoly(p1,p2,t1,t2,p3);
	printf(" \n\n Resultant polynomial after addition : ");
	displayPoly(p3,t3);
	printf("\n");
	return 0;
}
int readPoly(struct poly p[10])
{
	int t1,i;
	printf("\n\n Enter the total number of terms in the polynomial:");
	scanf("%d",&t1);
	printf("\n Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER\n");
	for(i=0;i<t1;i++)
	{
		printf("Enter the Coefficient(%d): ",i+1);
		scanf("%d",&p[i].coeff);
		printf("Enter the exponent(%d): ",i+1);
		scanf("%d",&p[i].expo); /* only statement in loop */
	}
	return(t1);
}
int addPoly(struct poly p1[10],struct poly p2[10],int t1,int t2,struct poly
p3[10])
{
	int i,j,k;
	i=0;
	j=0;
```

```
k=0;
while(i<t1 && j<t2)
{
	if(p1[i].expo==p2[j].expo)
	{
		p3[k].coeff=p1[i].coeff + p2[j].coeff;
		p3[k].expo=p1[i].expo;
		i++;
		j++;
		k++;
	}
	else if(p1[i].expo>p2[j].expo)
	{
		p3[k].coeff=p1[i].coeff;
		p3[k].expo=p1[i].expo;
		i++;
		k++;
	}
	else
	{
		p3[k].coeff=p2[j].coeff;
		p3[k].expo=p2[j].expo;
		j++;
		k++;
	}
}
/* for rest over terms of polynomial 1 */
while(i<t1)
{
	p3[k].coeff=p1[i].coeff;
	p3[k].expo=p1[i].expo;
	i++;
```

```
            k++;
      }
      /* for rest over terms of polynomial 2 */
      while(j<t2)
      {
            p3[k].coeff=p2[j].coeff;
            p3[k].expo=p2[j].expo;
            j++;
            k++;
      }
      return(k); /* k is number of terms in resultant polynomial*/
}
void displayPoly(struct poly p[10],int term)
{
      int k;
      for(k=0;k<term-1;k++)
            printf("%d(x^%d)+",p[k].coeff,p[k].expo);
      printf("%d(x^%d)",p[term-1].coeff,p[term-1].expo);
}
```

Sample Input and Output:

```
Enter the total number of terms in the polynomial: 3

Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER
  Enter the Coefficient(1): 7
     Enter the exponent(1): 2
  Enter the Coefficient(2): 4
     Enter the exponent(2): 1
  Enter the Coefficient(3): 6
     Enter the exponent(3): 0

First polynomial : 7(x^2)+4(x^1)+6(x^0)

Enter the total number of terms in the polynomial: 3

Enter the COEFFICIENT and EXPONENT in DESCENDING ORDER
  Enter the Coefficient(1): 9
     Enter the exponent(1): 2
  Enter the Coefficient(2): 8
     Enter the exponent(2): 1
  Enter the Coefficient(3): 5
     Enter the exponent(3): 0

Second polynomial : 9(x^2)+8(x^1)+5(x^0)

Resultant polynomial after addition : 16(x^2)+12(x^1)+11(x^0)

--------------------------------
Process exited after 43.24 seconds with return value 0
Press any key to continue . . .
```