DATA STRUCTURES

Division	**CS(AIML) -A**
Batch	**2**
GR-no	**12311493**
Roll no	**54**
Name	**Atharva Kangralkar**

Assignment 7:

Implement a Polynomial addition and multiplication using Linked Lists.

Code:-

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

typedef struct PolyNode {
    int coeff;
    int expo;
    struct PolyNode *next;
} PolyNode;

PolyNode* createNode(int coeff, int expo) {
    PolyNode* newNode = (PolyNode*)malloc(sizeof(PolyNode));
    newNode->coeff = coeff;
    newNode->expo = expo;
    newNode->next = NULL;
    return newNode;
}

void insertTerm(PolyNode** poly, int coeff, int expo) {
    if (coeff == 0) return;
    PolyNode* newNode = createNode(coeff, expo);

    if (*poly == NULL || (*poly)->expo < expo) {
        newNode->next = *poly;
        *poly = newNode;
    } else {
        PolyNode* temp = *poly;
        while (temp->next != NULL && temp->next->expo > expo)
            temp = temp->next;
```

```
        if (temp->next != NULL && temp->next->expo == expo) {
            temp->next->coeff += coeff;
            free(newNode);
        } else {
            newNode->next = temp->next;
            temp->next = newNode;
        }
    }
}

void displayPoly(PolyNode* poly) {
    while (poly != NULL) {
        printf("%dx^%d", poly->coeff, poly->expo);
        if (poly->next != NULL)
            printf(" + ");
        poly = poly->next;
    }
    printf("\n");
}

PolyNode* addPoly(PolyNode* p1, PolyNode* p2) {
    PolyNode* result = NULL;

    while (p1 != NULL && p2 != NULL) {
        if (p1->expo == p2->expo) {
            insertTerm(&result, p1->coeff + p2->coeff, p1->expo);
            p1 = p1->next;
            p2 = p2->next;
```

```
        } else if (p1->expo > p2->expo) {
            insertTerm(&result, p1->coeff, p1->expo);
            p1 = p1->next;
        } else {
            insertTerm(&result, p2->coeff, p2->expo);
            p2 = p2->next;
        }
    }

    while (p1 != NULL) {
        insertTerm(&result, p1->coeff, p1->expo);
        p1 = p1->next;
    }

    while (p2 != NULL) {
        insertTerm(&result, p2->coeff, p2->expo);
        p2 = p2->next;
    }

    return result;
}

PolyNode* multiplyPoly(PolyNode* p1, PolyNode* p2) {
    PolyNode* result = NULL;

    for (PolyNode* ptr1 = p1; ptr1 != NULL; ptr1 = ptr1->next) {
        for (PolyNode* ptr2 = p2; ptr2 != NULL; ptr2 = ptr2->next) {
            int coeff = ptr1->coeff * ptr2->coeff;
            int expo = ptr1->expo + ptr2->expo;
```

```
            insertTerm(&result, coeff, expo);
        }
    }

    return result;
}

void freePoly(PolyNode* poly) {
    while (poly != NULL) {
        PolyNode* temp = poly;
        poly = poly->next;
        free(temp);
    }
}

void inputPoly(PolyNode** poly) {
    int n, coeff, expo;
    printf("Enter number of terms: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        printf("Enter coefficient and exponent for term %d: ", i + 1);
        scanf("%d %d", &coeff, &expo);
        insertTerm(poly, coeff, expo);
    }
}

int main() {
    PolyNode *p1 = NULL, *p2 = NULL, *sum = NULL, *product = NULL;
```

```
    printf("Enter Polynomial 1:\n");
    inputPoly(&p1);
    printf("Enter Polynomial 2:\n");
    inputPoly(&p2);

    printf("\nPolynomial 1: ");
    displayPoly(p1);
    printf("Polynomial 2: ");
    displayPoly(p2);

    sum = addPoly(p1, p2);
    printf("\nSum: ");
    displayPoly(sum);

    product = multiplyPoly(p1, p2);
    printf("Product: ");
    displayPoly(product);

    freePoly(p1);
    freePoly(p2);
    freePoly(sum);
    freePoly(product);

    return 0;
}
```

Code Screenshot:-

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

typedef struct PolyNode {
    int coeff;
    int expo;
    struct PolyNode *next;
} PolyNode;

PolyNode* createNode(int coeff, int expo) {
    PolyNode* newNode = (PolyNode*)malloc(sizeof(PolyNode));
    newNode->coeff = coeff;
    newNode->expo = expo;
    newNode->next = NULL;
    return newNode;
}

void insertTerm(PolyNode** poly, int coeff, int expo) {
    if (coeff == 0) return;
    PolyNode* newNode = createNode(coeff, expo);

    if (*poly == NULL || (*poly)->expo < expo) {
        newNode->next = *poly;
        *poly = newNode;
    } else {
        PolyNode* temp = *poly;
```

```
        PolyNode* temp = *poly;
        while (temp->next != NULL && temp->next->expo > expo)
            temp = temp->next;

        if (temp->next != NULL && temp->next->expo == expo) {
            temp->next->coeff += coeff;
            free(newNode);
        } else {
            newNode->next = temp->next;
            temp->next = newNode;
        }
    }
}

void displayPoly(PolyNode* poly) {
    while (poly != NULL) {
        printf("%dx^%d", poly->coeff, poly->expo);
        if (poly->next != NULL)
            printf(" + ");
        poly = poly->next;
    }
    printf("\n");
}

PolyNode* addPoly(PolyNode* p1, PolyNode* p2) {
```

```
    while (p1 != NULL && p2 != NULL) {
        if (p1->expo == p2->expo) {
            insertTerm(&result, p1->coeff + p2->coeff, p1->expo);
            p1 = p1->next;
            p2 = p2->next;
        } else if (p1->expo > p2->expo) {
            insertTerm(&result, p1->coeff, p1->expo);
            p1 = p1->next;
        } else {
            insertTerm(&result, p2->coeff, p2->expo);
            p2 = p2->next;
        }
    }

    while (p1 != NULL) {
        insertTerm(&result, p1->coeff, p1->expo);
        p1 = p1->next;
    }

    while (p2 != NULL) {
        insertTerm(&result, p2->coeff, p2->expo);
        p2 = p2->next;
    }

    return result;
}
```

```
    return result;
}

PolyNode* multiplyPoly(PolyNode* p1, PolyNode* p2) {
    PolyNode* result = NULL;

    for (PolyNode* ptr1 = p1; ptr1 != NULL; ptr1 = ptr1->next) {
        for (PolyNode* ptr2 = p2; ptr2 != NULL; ptr2 = ptr2->next)
            {
            int coeff = ptr1->coeff * ptr2->coeff;
            int expo = ptr1->expo + ptr2->expo;
            insertTerm(&result, coeff, expo);
        }
    }

    return result;
}

void freePoly(PolyNode* poly) {
    while (poly != NULL) {
        PolyNode* temp = poly;
        poly = poly->next;
        free(temp);
    }
}
```

```c
void inputPoly(PolyNode** poly) {
    int n, coeff, expo;
    printf("Enter number of terms: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        printf("Enter coefficient and exponent for term %d: ", i +
            1);
        scanf("%d %d", &coeff, &expo);
        insertTerm(poly, coeff, expo);
    }
}

int main() {
    PolyNode *p1 = NULL, *p2 = NULL, *sum = NULL, *product = NULL;

    printf("Enter Polynomial 1:\n");
    inputPoly(&p1);
    printf("Enter Polynomial 2:\n");
    inputPoly(&p2);

    printf("\nPolynomial 1: ");
    displayPoly(p1);
    printf("Polynomial 2: ");
    displayPoly(p2);
```

```c
int main() {
    PolyNode *p1 = NULL, *p2 = NULL, *sum = NULL, *product = NULL;

    printf("Enter Polynomial 1:\n");
    inputPoly(&p1);
    printf("Enter Polynomial 2:\n");
    inputPoly(&p2);

    printf("\nPolynomial 1: ");
    displayPoly(p1);
    printf("Polynomial 2: ");
    displayPoly(p2);

    sum = addPoly(p1, p2);
    printf("\nSum: ");
    displayPoly(sum);

    product = multiplyPoly(p1, p2);
    printf("Product: ");
    displayPoly(product);

    freePoly(p1);
    freePoly(p2);
    freePoly(sum);
    freePoly(product);
```

```c
    product = multiplyPoly(p1, p2);
    printf("Product: ");
    displayPoly(product);

    freePoly(p1);
    freePoly(p2);
    freePoly(sum);
    freePoly(product);

    return 0;
}
```

Output:-

```
Enter Polynomial 1:
Enter number of terms: 3
Enter coefficient and exponent for term 1: 2 2
Enter coefficient and exponent for term 2: 3 1
Enter coefficient and exponent for term 3: 5 0
Enter Polynomial 2:
Enter number of terms: 2
Enter coefficient and exponent for term 1: 1 6
Enter coefficient and exponent for term 2: 3 2

Polynomial 1: 2x^2 + 3x^1 + 5x^0
Polynomial 2: 1x^6 + 3x^2

Sum: 1x^6 + 5x^2 + 3x^1 + 5x^0
Product: 2x^8 + 3x^7 + 5x^6 + 6x^4 + 9x^3 + 15x^2
```