Polynomial

-Aaryan Sharma
-16010123012

Single variable -

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

struct Term {
    int coefficient;
    int exponent;
};

struct Polynomial {
    struct Term *terms;
    int numTerms;
};

struct Term *createTerm(int coeff, int exp) {
    struct Term *newTerm = (struct Term *)malloc(sizeof(struct Term));
    newTerm->coefficient = coeff;
    newTerm->exponent = exp;
    return newTerm;
}

struct Polynomial *createPolynomial(int numTerms) {
```

```
    struct Polynomial *newPoly = (struct Polynomial *)malloc(sizeof(struct
Polynomial));
    newPoly->numTerms = numTerms;
    newPoly->terms = (struct Term *)malloc(numTerms * sizeof(struct Term));
    return newPoly;
}

struct Polynomial *addPolynomials(struct Polynomial *poly1, struct Polynomial
*poly2) {
    int i, j, k = 0;
    struct Polynomial *resultPoly = createPolynomial(poly1->numTerms + poly2-
>numTerms);
    i = j = 0;

    while (i < poly1->numTerms && j < poly2->numTerms) {
        if (poly1->terms[i].exponent > poly2->terms[j].exponent) {
            resultPoly->terms[k++] = poly1->terms[i++];
        } else if (poly1->terms[i].exponent < poly2->terms[j].exponent) {
            resultPoly->terms[k++] = poly2->terms[j++];
        } else {
            int sumCoeff = poly1->terms[i].coefficient + poly2->terms[j].coefficient;
            if (sumCoeff != 0) {
                resultPoly->terms[k].coefficient = sumCoeff;
                resultPoly->terms[k].exponent = poly1->terms[i].exponent;
                k++;
```

```
            }
            i++;
            j++;
        }
    }

    while (i < poly1->numTerms) {
        resultPoly->terms[k++] = poly1->terms[i++];
    }

    while (j < poly2->numTerms) {
        resultPoly->terms[k++] = poly2->terms[j++];
    }

    resultPoly->numTerms = k;
    return resultPoly;
}

void printPolynomial(struct Polynomial *poly) {
    for (int i = 0; i < poly->numTerms; i++) {
        if (poly->terms[i].coefficient != 0) {
            printf("%dx^%d", poly->terms[i].coefficient, poly->terms[i].exponent);
            if (i < poly->numTerms - 1) {
                printf(" + ");
```

```
            }
        }
    }
    printf("\n");
}

int main() {
    struct Polynomial *poly1 = createPolynomial(4);
    poly1->terms[0] = *createTerm(10, 3);
    poly1->terms[1] = *createTerm(-1, 2);
    poly1->terms[2] = *createTerm(5, 1);
    poly1->terms[3] = *createTerm(-7, 0);

    struct Polynomial *poly2 = createPolynomial(3);
    poly2->terms[0] = *createTerm(4, 2);
    poly2->terms[1] = *createTerm(6, 1);
    poly2->terms[2] = *createTerm(9, 0);

    struct Polynomial *resultPoly = addPolynomials(poly1, poly2);

    printf("Polynomial 1: ");
    printPolynomial(poly1);

    printf("Polynomial 2: ");
```

```
    printPolynomial(poly2);

    printf("Resultant Polynomial: ");
    printPolynomial(resultPoly);

    free(poly1->terms);
    free(poly1);
    free(poly2->terms);
    free(poly2);
    free(resultPoly->terms);
    free(resultPoly);

    return 0;
}
```

```
Polynomial 1: 10x^3 + -1x^2 + 5x^1 + -7x^0
Polynomial 2: 4x^2 + 6x^1 + 9x^0
Resultant Polynomial: 10x^3 + 3x^2 + 11x^1 + 2x^0


=== Code Execution Successful ===
```

Three variable –

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

```
struct Node {
   int coef;
   int type;
   int exp;
   struct Node* next;
};

typedef struct Node Node;

void insert(Node** poly, int coef, int type, int exp) {
   Node* temp = (Node*)malloc(sizeof(Node));
   temp->coef = coef;
   temp->type = type;
   temp->exp = exp;
   temp->next = NULL;

   if (*poly == NULL || ((*poly)->type > type || ((*poly)->type == type && (*poly)->exp > exp))) {
      temp->next = *poly;
      *poly = temp;
      return;
   }

   Node* current = *poly;
   Node* prev = NULL;
```

```
   while (current != NULL && (current->type < type || (current->type == type && current->exp < exp))) {
      prev = current;
      current = current->next;
   }

   if (current != NULL && current->type == type && current->exp == exp) {
      current->coef += coef;
      if (current->coef == 0) {
         if (prev != NULL) {
            prev->next = current->next;
         } else {
            *poly = current->next;
         }
         free(current);
      }
      free(temp); // Free the temporary node as it was not used
   } else {
      if (prev == NULL) {
         temp->next = *poly;
         *poly = temp;
      } else {
         prev->next = temp;
         temp->next = current;
```

```
        }
    }
}

void print(Node* poly) {
    if (poly == NULL) {
        printf("0\n");
        return;
    }

    Node* current = poly;
    int first = 1;

    while (current != NULL) {
        if (current->coef != 0) {
            if (first) {
                first = 0;
            } else {
                printf(" + ");
            }

            switch (current->type) {
                case 1: printf("%dx^%d", current->coef, current->exp); break;
                case 2: printf("%dy^%d", current->coef, current->exp); break;
```

```
            case 3: printf("%dz^%d", current->coef, current->exp); break;
            case 4: printf("%d", current->coef); break;
        }
      }
      current = current->next;
   }

   printf("\n");
}

Node* add(Node* poly1, Node* poly2) {
   Node* result = NULL;

   while (poly1 != NULL && poly2 != NULL) {
      if (poly1->type == poly2->type && poly1->exp == poly2->exp) {
         insert(&result, poly1->coef + poly2->coef, poly1->type, poly1->exp);
         poly1 = poly1->next;
         poly2 = poly2->next;
      } else if (poly1->type < poly2->type || (poly1->type == poly2->type && poly1->exp < poly2->exp)) {
         insert(&result, poly1->coef, poly1->type, poly1->exp);
         poly1 = poly1->next;
      } else {
         insert(&result, poly2->coef, poly2->type, poly2->exp);
         poly2 = poly2->next;
```

```
        }
    }

    while (poly1 != NULL) {
        insert(&result, poly1->coef, poly1->type, poly1->exp);
        poly1 = poly1->next;
    }

    while (poly2 != NULL) {
        insert(&result, poly2->coef, poly2->type, poly2->exp);
        poly2 = poly2->next;
    }

    return result;
}

int main() {
    Node* poly1 = NULL;
    insert(&poly1, 5, 1, 4);
    insert(&poly1, 7, 2, 3);
    insert(&poly1, 3, 3, 2);
    insert(&poly1, 9, 4, 1);

    Node* poly2 = NULL;
```

```
    insert(&poly2, -2, 1, 4);
    insert(&poly2, -4, 2, 3);
    insert(&poly2, -6, 3, 2);

    printf("First polynomial: ");
    print(poly1);

    printf("Second polynomial: ");
    print(poly2);

    Node* result = add(poly1, poly2);
    printf("Result: ");
    print(result);

    return 0;
}
```

```
First polynomial: 5x^4 + 7y^3 + 3z^2 + 9
Second polynomial: -2x^4 + -4y^3 + -6z^2
Result: 3x^4 + 3y^3 + -3z^2 + 9


=== Code Execution Successful ===
```