DS LAB 3

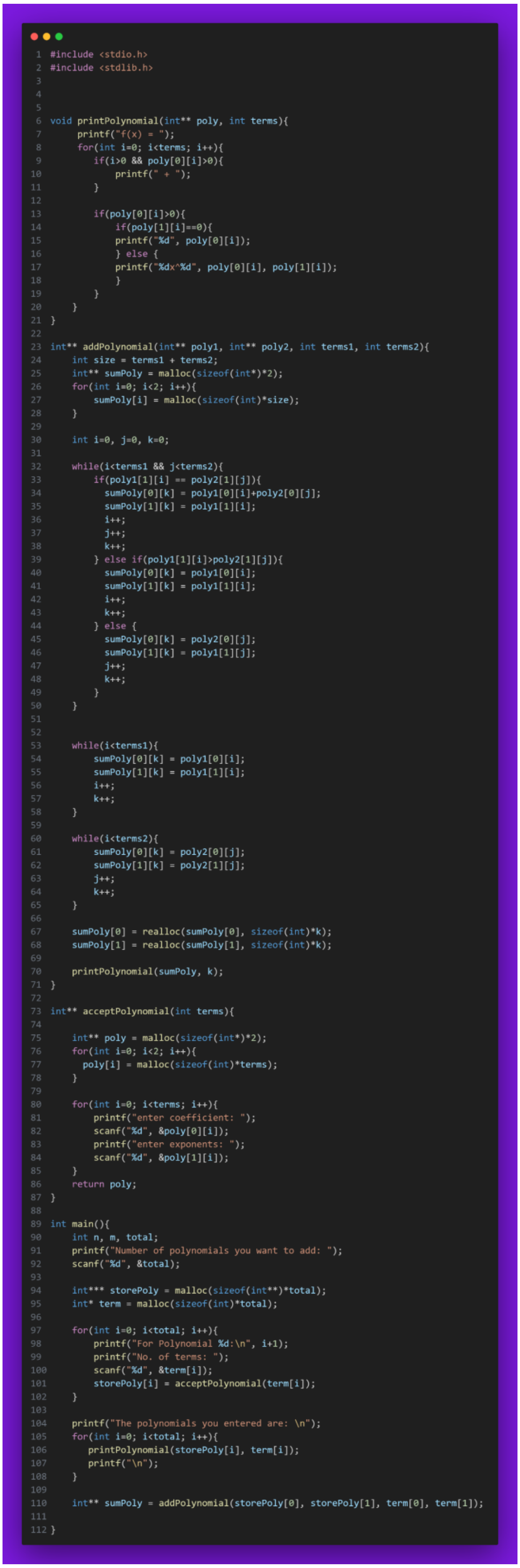
12/08/2024

Name: Utkarsh Yadav

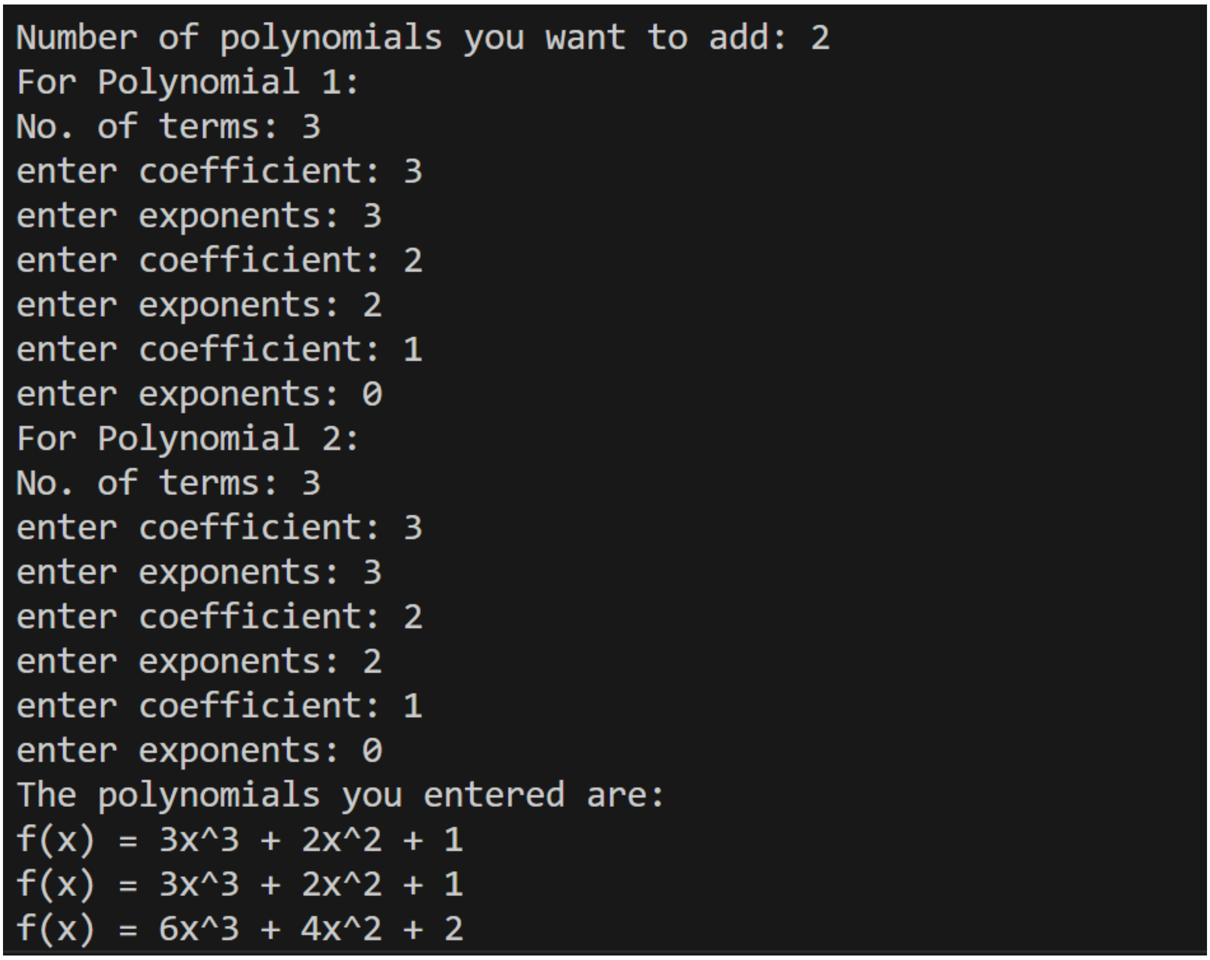
Roll: 23053172

Sec: CSE37

ADDITION OF POLYNOMIALS

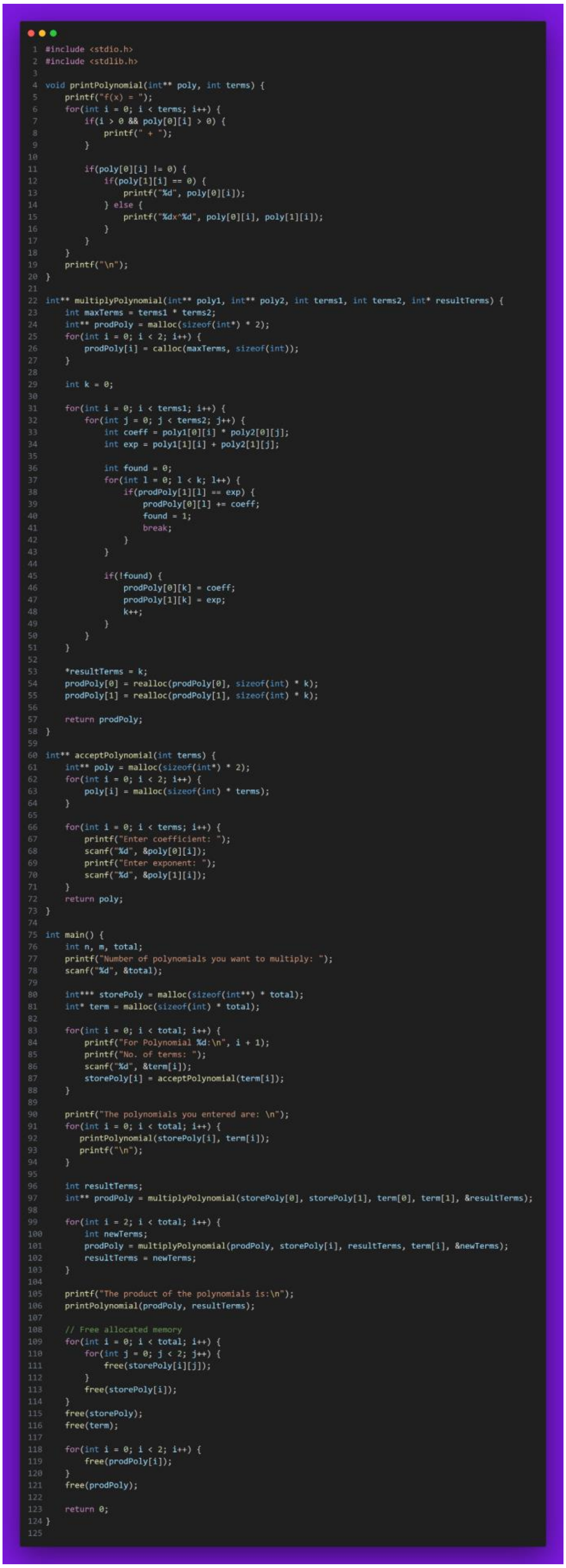


```c
#include <stdio.h>
#include <stdlib.h>



void printPolynomial(int** poly, int terms){
     printf("f(x) = ");
     for(int i=0; i<terms; i++){
        if(i>0 && poly[0][i]>0){
            printf(" + ");
        }

        if(poly[0][i]>0){
            if(poly[1][i]==0){
            printf("%d", poly[0][i]);
            } else {
            printf("%dx^%d", poly[0][i], poly[1][i]);
            }
        }
    }
}

int** addPolynomial(int** poly1, int** poly2, int terms1, int terms2){
    int size = terms1 + terms2;
    int** sumPoly = malloc(sizeof(int*)*2);
    for(int i=0; i<2; i++){
        sumPoly[i] = malloc(sizeof(int)*size);
    }

    int i=0, j=0, k=0;

    while(i<terms1 && j<terms2){
        if(poly1[1][i] == poly2[1][j]){
          sumPoly[0][k] = poly1[0][i]+poly2[0][j];
          sumPoly[1][k] = poly1[1][i];
          i++;
          j++;
          k++;
        } else if(poly1[1][i]>poly2[1][j]){
          sumPoly[0][k] = poly1[0][i];
          sumPoly[1][k] = poly1[1][i];
          i++;
          k++;
        } else {
          sumPoly[0][k] = poly2[0][j];
          sumPoly[1][k] = poly1[1][j];
          j++;
          k++;
        }
    }


    while(i<terms1){
        sumPoly[0][k] = poly1[0][i];
        sumPoly[1][k] = poly1[1][i];
        i++;
        k++;
    }

    while(i<terms2){
        sumPoly[0][k] = poly2[0][j];
        sumPoly[1][k] = poly2[1][j];
        j++;
        k++;
    }

    sumPoly[0] = realloc(sumPoly[0], sizeof(int)*k);
    sumPoly[1] = realloc(sumPoly[1], sizeof(int)*k);

    printPolynomial(sumPoly, k);
}

int** acceptPolynomial(int terms){

    int** poly = malloc(sizeof(int*)*2);
    for(int i=0; i<2; i++){
      poly[i] = malloc(sizeof(int)*terms);
    }

    for(int i=0; i<terms; i++){
        printf("enter coefficient: ");
        scanf("%d", &poly[0][i]);
        printf("enter exponents: ");
        scanf("%d", &poly[1][i]);
    }
    return poly;
}

int main(){
    int n, m, total;
    printf("Number of polynomials you want to add: ");
    scanf("%d", &total);

    int*** storePoly = malloc(sizeof(int**)*total);
    int* term = malloc(sizeof(int)*total);

    for(int i=0; i<total; i++){
        printf("For Polynomial %d:\n", i+1);
        printf("No. of terms: ");
        scanf("%d", &term[i]);
        storePoly[i] = acceptPolynomial(term[i]);
    }

    printf("The polynomials you entered are: \n");
    for(int i=0; i<total; i++){
       printPolynomial(storePoly[i], term[i]);
       printf("\n");
    }

    int** sumPoly = addPolynomial(storePoly[0], storePoly[1], term[0], term[1]);

}
```

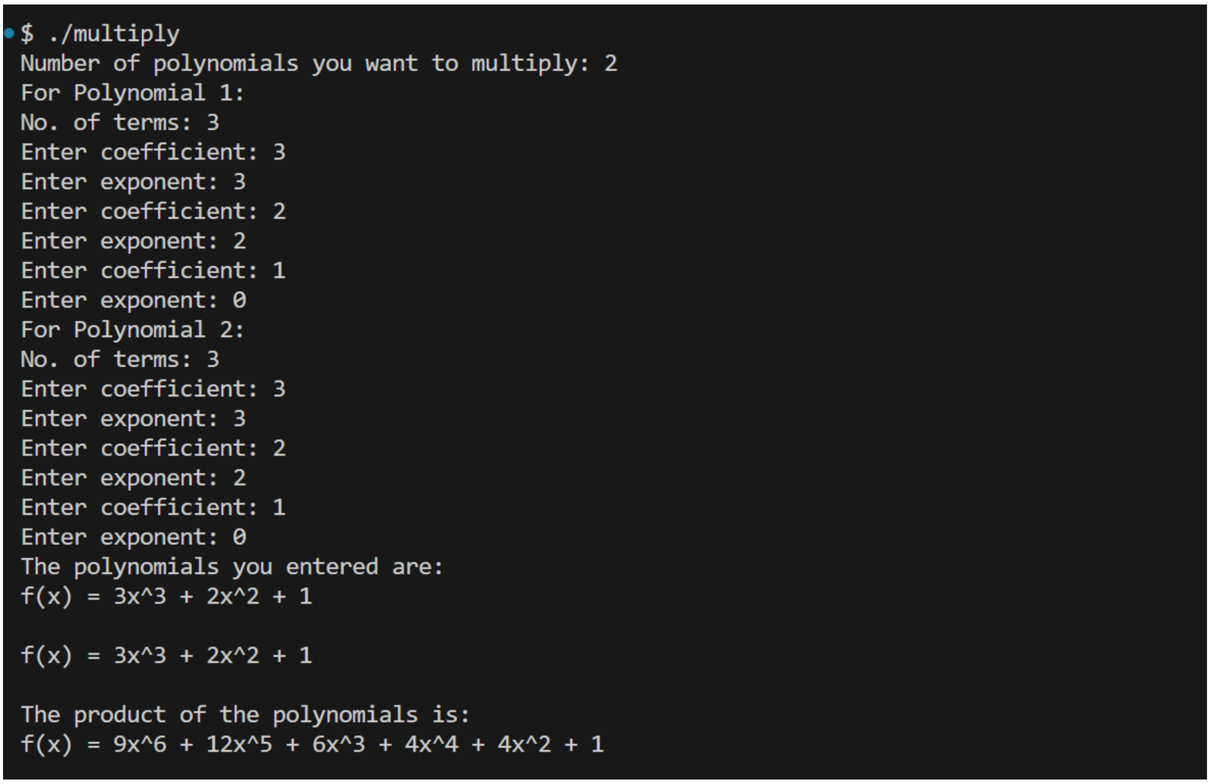


```
Number of polynomials you want to add: 2
For Polynomial 1:
No. of terms: 3
enter coefficient: 3
enter exponents: 3
enter coefficient: 2
enter exponents: 2
enter coefficient: 1
enter exponents: 0
For Polynomial 2:
No. of terms: 3
enter coefficient: 3
enter exponents: 3
enter coefficient: 2
enter exponents: 2
enter coefficient: 1
enter exponents: 0
The polynomials you entered are:
f(x) = 3x^3 + 2x^2 + 1
f(x) = 3x^3 + 2x^2 + 1
f(x) = 6x^3 + 4x^2 + 2
```

MULTIPLICATION OF POLYNOMIALS

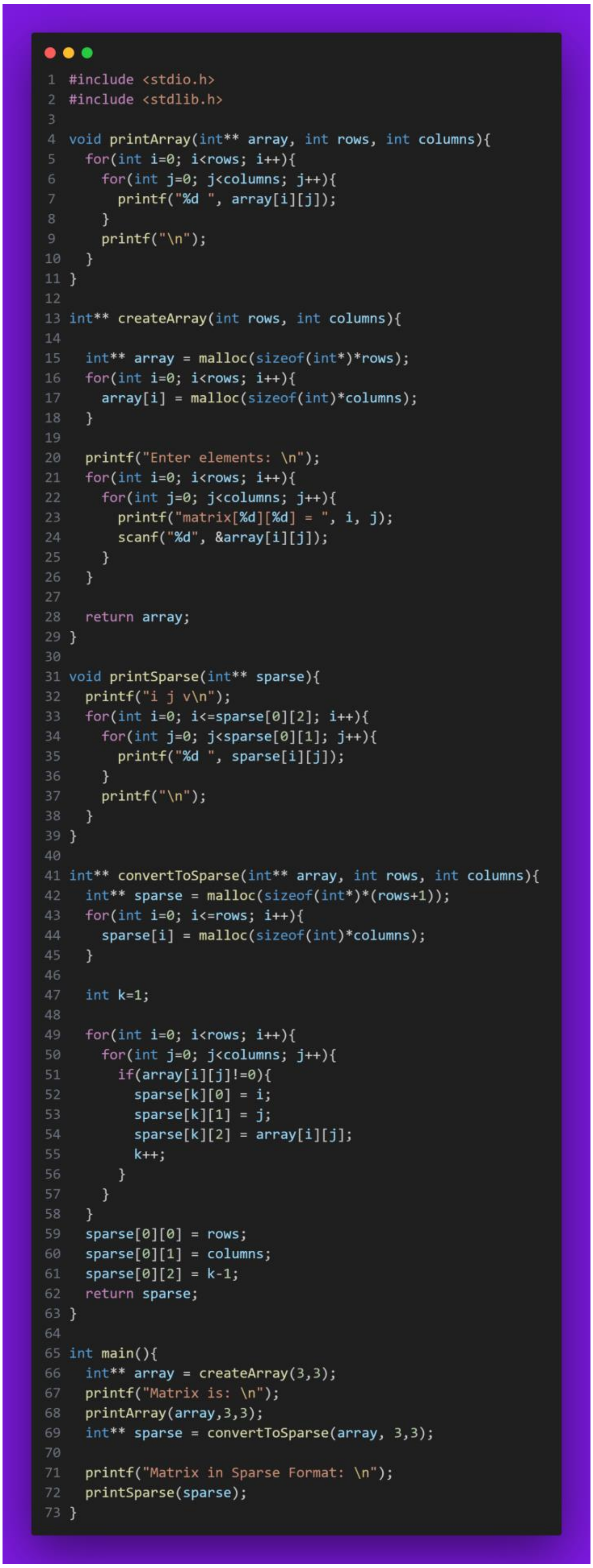


```c
#include <stdio.h>
#include <stdlib.h>

void printPolynomial(int** poly, int terms) {
    printf("f(x) = ");
    for(int i = 0; i < terms; i++) {
        if(i > 0 && poly[0][i] > 0) {
            printf(" + ");
        }

        if(poly[0][i] != 0) {
            if(poly[1][i] == 0) {
                printf("%d", poly[0][i]);
            } else {
                printf("%dx^%d", poly[0][i], poly[1][i]);
            }
        }
    }
    printf("\n");
}

int** multiplyPolynomial(int** poly1, int** poly2, int terms1, int terms2, int* resultTerms) {
    int maxTerms = terms1 * terms2;
    int** prodPoly = malloc(sizeof(int*) * 2);
    for(int i = 0; i < 2; i++) {
        prodPoly[i] = calloc(maxTerms, sizeof(int));
    }

    int k = 0;

    for(int i = 0; i < terms1; i++) {
        for(int j = 0; j < terms2; j++) {
            int coeff = poly1[0][i] * poly2[0][j];
            int exp = poly1[1][i] + poly2[1][j];

            int found = 0;
            for(int l = 0; l < k; l++) {
                if(prodPoly[1][l] == exp) {
                    prodPoly[0][l] += coeff;
                    found = 1;
                    break;
                }
            }

            if(!found) {
                prodPoly[0][k] = coeff;
                prodPoly[1][k] = exp;
                k++;
            }
        }
    }

    *resultTerms = k;
    prodPoly[0] = realloc(prodPoly[0], sizeof(int) * k);
    prodPoly[1] = realloc(prodPoly[1], sizeof(int) * k);

    return prodPoly;
}

int** acceptPolynomial(int terms) {
    int** poly = malloc(sizeof(int*) * 2);
    for(int i = 0; i < 2; i++) {
        poly[i] = malloc(sizeof(int) * terms);
    }

    for(int i = 0; i < terms; i++) {
        printf("Enter coefficient: ");
        scanf("%d", &poly[0][i]);
        printf("Enter exponent: ");
        scanf("%d", &poly[1][i]);
    }
    return poly;
}

int main() {
    int n, m, total;
    printf("Number of polynomials you want to multiply: ");
    scanf("%d", &total);

    int*** storePoly = malloc(sizeof(int**) * total);
    int* term = malloc(sizeof(int) * total);

    for(int i = 0; i < total; i++) {
        printf("For Polynomial %d:\n", i + 1);
        printf("No. of terms: ");
        scanf("%d", &term[i]);
        storePoly[i] = acceptPolynomial(term[i]);
    }

    printf("The polynomials you entered are: \n");
    for(int i = 0; i < total; i++) {
       printPolynomial(storePoly[i], term[i]);
       printf("\n");
    }

    int resultTerms;
    int** prodPoly = multiplyPolynomial(storePoly[0], storePoly[1], term[0], term[1], &resultTerms);

    for(int i = 2; i < total; i++) {
        int newTerms;
        prodPoly = multiplyPolynomial(prodPoly, storePoly[i], resultTerms, term[i], &newTerms);
        resultTerms = newTerms;
    }

    printf("The product of the polynomials is:\n");
    printPolynomial(prodPoly, resultTerms);

    // Free allocated memory
    for(int i = 0; i < total; i++) {
        for(int j = 0; j < 2; j++) {
            free(storePoly[i][j]);
        }
        free(storePoly[i]);
    }
    free(storePoly);
    free(term);

    for(int i = 0; i < 2; i++) {
        free(prodPoly[i]);
    }
    free(prodPoly);

    return 0;
}
```

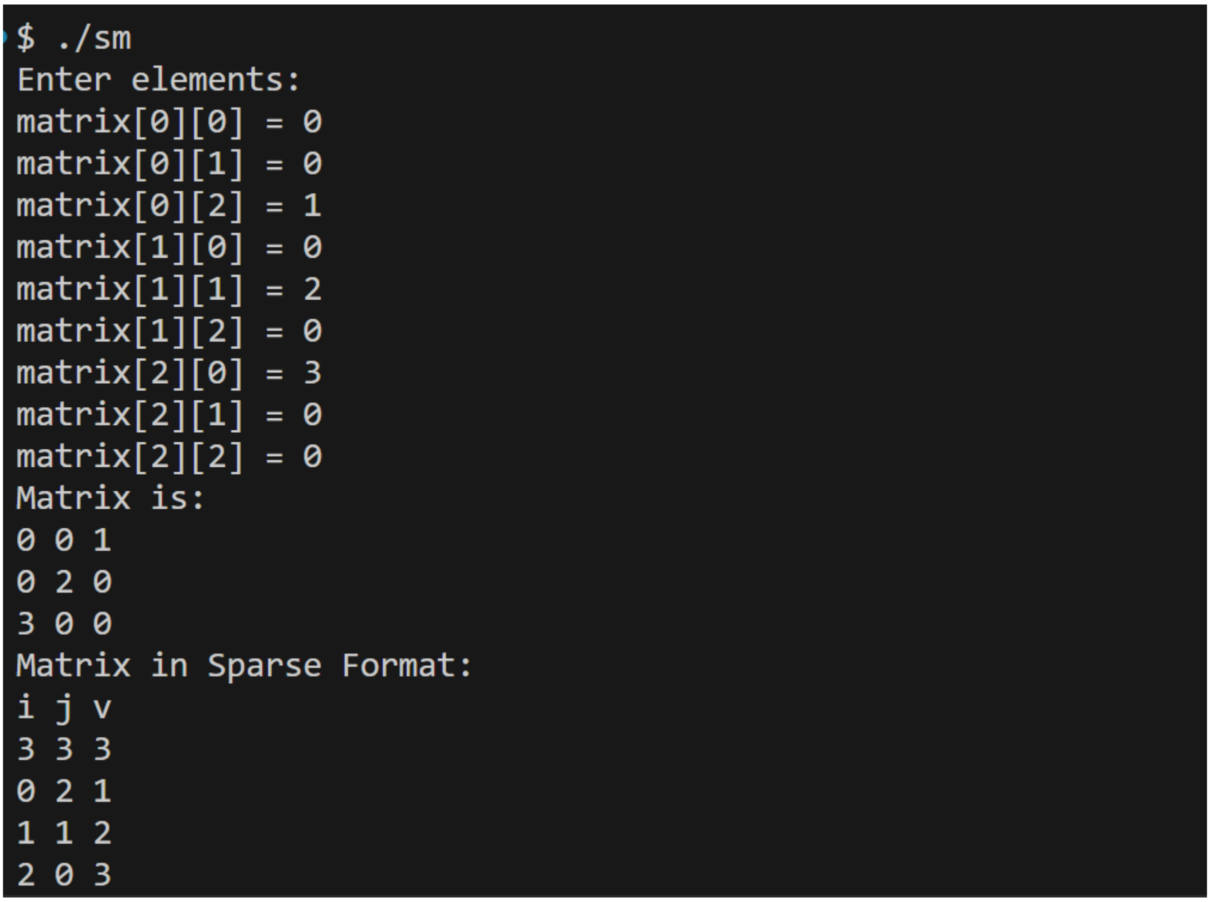


```
$ ./multiply
Number of polynomials you want to multiply: 2
For Polynomial 1:
No. of terms: 3
Enter coefficient: 3
Enter exponent: 3
Enter coefficient: 2
Enter exponent: 2
Enter coefficient: 1
Enter exponent: 0
For Polynomial 2:
No. of terms: 3
Enter coefficient: 3
Enter exponent: 3
Enter coefficient: 2
Enter exponent: 2
Enter coefficient: 1
Enter exponent: 0
The polynomials you entered are:
f(x) = 3x^3 + 2x^2 + 1

f(x) = 3x^3 + 2x^2 + 1

The product of the polynomials is:
f(x) = 9x^6 + 12x^5 + 6x^3 + 4x^4 + 4x^2 + 1
```

IMPLEMENT SPARSE MATRIX



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

void printArray(int** array, int rows, int columns){
  for(int i=0; i<rows; i++){
    for(int j=0; j<columns; j++){
      printf("%d ", array[i][j]);
    }
    printf("\n");
  }
}

int** createArray(int rows, int columns){

  int** array = malloc(sizeof(int*)*rows);
  for(int i=0; i<rows; i++){
    array[i] = malloc(sizeof(int)*columns);
  }

  printf("Enter elements: \n");
  for(int i=0; i<rows; i++){
    for(int j=0; j<columns; j++){
      printf("matrix[%d][%d] = ", i, j);
      scanf("%d", &array[i][j]);
    }
  }

  return array;
}

void printSparse(int** sparse){
  printf("i j v\n");
  for(int i=0; i<=sparse[0][2]; i++){
    for(int j=0; j<sparse[0][1]; j++){
      printf("%d ", sparse[i][j]);
    }
    printf("\n");
  }
}

int** convertToSparse(int** array, int rows, int columns){
  int** sparse = malloc(sizeof(int*)*(rows+1));
  for(int i=0; i<=rows; i++){
    sparse[i] = malloc(sizeof(int)*columns);
  }

  int k=1;

  for(int i=0; i<rows; i++){
    for(int j=0; j<columns; j++){
      if(array[i][j]!=0){
        sparse[k][0] = i;
        sparse[k][1] = j;
        sparse[k][2] = array[i][j];
        k++;
      }
    }
  }
  sparse[0][0] = rows;
  sparse[0][1] = columns;
  sparse[0][2] = k-1;
  return sparse;
}

int main(){
  int** array = createArray(3,3);
  printf("Matrix is: \n");
  printArray(array,3,3);
  int** sparse = convertToSparse(array, 3,3);

  printf("Matrix in Sparse Format: \n");
  printSparse(sparse);
}
```



```
$ ./sm
Enter elements:
matrix[0][0] = 0
matrix[0][1] = 0
matrix[0][2] = 1
matrix[1][0] = 0
matrix[1][1] = 2
matrix[1][2] = 0
matrix[2][0] = 3
matrix[2][1] = 0
matrix[2][2] = 0
Matrix is:
0 0 1
0 2 0
3 0 0
Matrix in Sparse Format:
i j v
3 3 3
0 2 1
1 1 2
2 0 3
```