Multiplication of Two Sparse Matrices

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

int** createMatrix(int rows, int cols){
  int ** array = (int**)malloc(rows*sizeof(int*));

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

void fillArray(int** array, int rows, int cols){
  printf("Enter Elements: \n");
  for(int i=0; i<rows; i++){
    for(int j=0; j<cols; j++){
      printf("Element [%d][%d]: ", i, j);
      scanf("%d", &array[i][j]);
    }
  }
}

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

void createSparse(int** array, int rows, int cols, int sparseMatrix[20][3]){
  int k=1;
  for(int i=0; i<rows; i++){
    for(int j=0; j<cols; j++){
      if(array[i][j] != 0){
        sparseMatrix[k][0] = i;
        sparseMatrix[k][1] = j;
        sparseMatrix[k][2] = array[i][j];
        k++;
      }
    }
  }
  sparseMatrix[0][0] = rows;
  sparseMatrix[0][1] = cols;
  sparseMatrix[0][2] = k-1;
}

void printSparse(int sparseMatrix[20][3]){
  printf("i\tj\tv\t\n");
  for(int i=0; i<sparseMatrix[0][2]+1; i++){
    printf("%d\t%d\t%d\t\n", sparseMatrix[i][0], sparseMatrix[i][1], sparseMatrix[i][2]);
  }
  printf("\n");
}
```

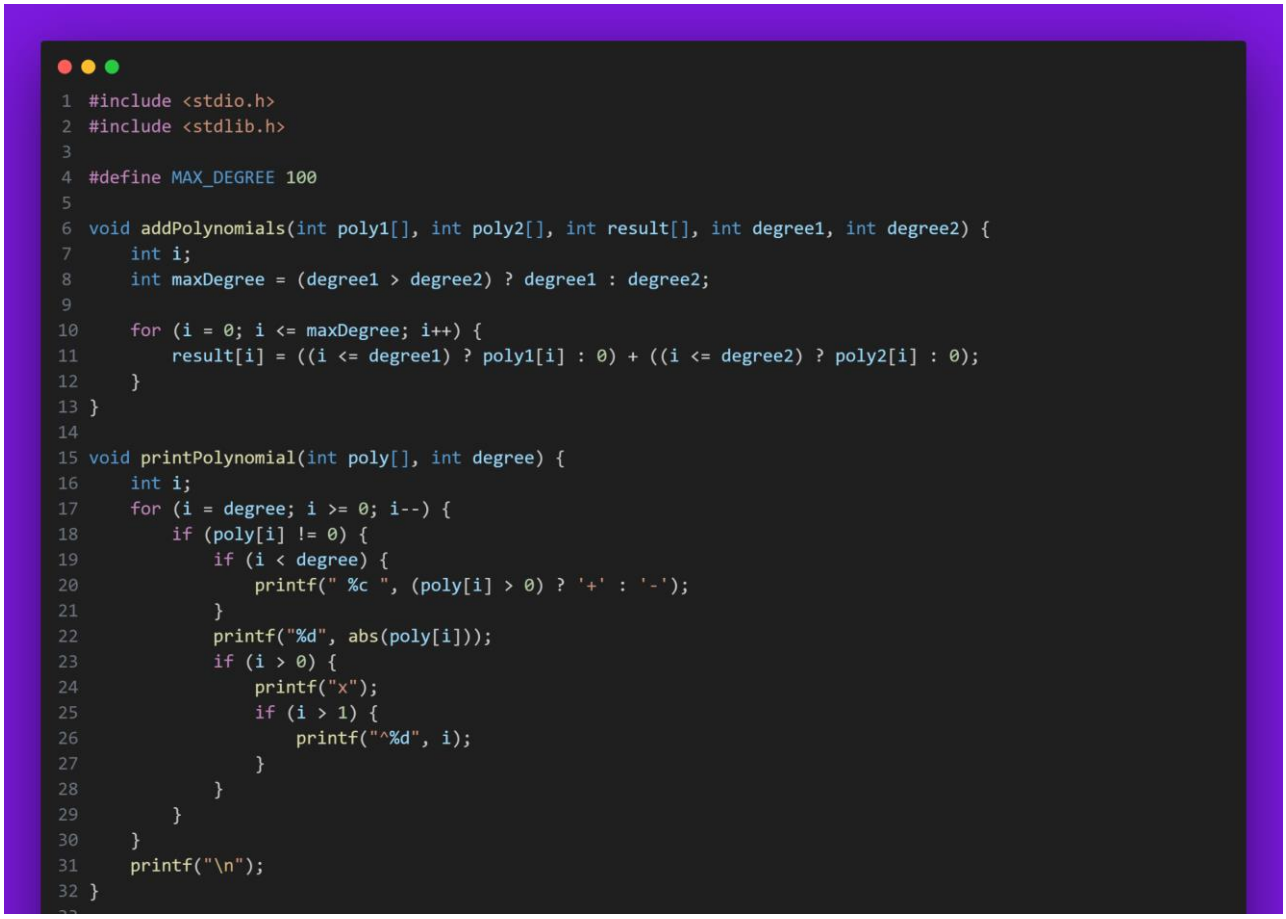
```c

void multiplySparse(int aSparse[20][3], int bSparse[20][3], int cSparse[20][3]){
  if(aSparse[0][1] != bSparse[0][0]){
    printf("Incompatible dimension");
    return;
  }

  int k=1;
  cSparse[0][0] = aSparse[0][0];
  cSparse[0][1] = bSparse[0][1];

  for(int i=1; i<= aSparse[0][2]; i++){
    for(int j=1; j<= bSparse[0][2]; j++){
      if(aSparse[i][1] == bSparse[j][0]){
        int row = aSparse[i][0];
        int col = bSparse[j][1];
        int value = aSparse[i][2]*bSparse[j][2];

        int found = 0;
        for(int x=1; x<k; x++){
          if(cSparse[x][0] == row && cSparse[x][1] == col){
            cSparse[x][2] += value;
            found = 1;
            break;
          }
        }
        if(!found){
          cSparse[k][0] = row;
          cSparse[k][1] = col;
          cSparse[k][2] = value;
          k++;
        }
      }
    }
  }
   cSparse[0][2] = k - 1;
}


int** convertSparseToNormal(int sparseMatrix[20][3]) {
    int rows = sparseMatrix[0][0];
    int cols = sparseMatrix[0][1];

    int** normalMatrix = (int**)calloc(rows, sizeof(int*));
    for (int i = 0; i < rows; i++) {
        normalMatrix[i] = (int*)calloc(cols, sizeof(int));
    }

    for (int i = 1; i <= sparseMatrix[0][2]; i++) {
        int row = sparseMatrix[i][0];
        int col = sparseMatrix[i][1];
        int value = sparseMatrix[i][2];
        normalMatrix[row][col] = value;
    }

    return normalMatrix;
}
```

```c

void freeMemory(int** array, int rows) {
    for (int i = 0; i < rows; i++) {
        free(array[i]);
    }
    free(array);
}

int main(){
  int aRow, aCol, bRow, bCol, cRow, cCol;
  int aSparse[20][3], bSparse[20][3], cSparse[20][3];

  printf("Matrix A: \n");
  printf("Rows and Columns: ");
  scanf("%d %d", &aRow, &aCol);

  int** a = createMatrix(aRow, aCol);
  fillArray(a, aRow, aCol);

  createSparse(a,aRow, aCol, aSparse);

  printf("Matrix B: \n");
  printf("Rows and Columns: ");
  scanf("%d %d", &bRow, &bCol);

  int** b = createMatrix(bRow, bCol);
  fillArray(b, bRow, bCol);
  createSparse(b,bRow, bCol, bSparse);



  printf("Matrix A is: \n");
  printMatrix(a, aRow, aCol);
  printf("Matrix A in Sparse Format: \n");
  printSparse(aSparse);

  printf("Matrix B is: \n");
  printMatrix(b, bRow, bCol);
  printf("Matrix B in Sparse Format: \n");
  printSparse(bSparse);

  multiplySparse(aSparse, bSparse, cSparse);
  printf("Resultant Matrix C in Sparse Format: \n");
  printSparse(cSparse);

   int** cNormal = convertSparseToNormal(cSparse);
    printf("Resultant Matrix C in Normal Format: \n");
    printMatrix(cNormal, cSparse[0][0], cSparse[0][1]);

    freeMemory(a, aRow);
    freeMemory(b, bRow);
    freeMemory(cNormal, cSparse[0][0]);

}
```

```
Matrix A is:
0 0 1
0 2 0
3 0 0

Matrix A in Sparse Format:
i       j       v
3       3       3
0       2       1
1       1       2
2       0       3

Matrix B is:
1 0 0
0 2 0
0 0 3

Matrix B in Sparse Format:
i       j       v
3       3       3
0       0       1
1       1       2
2       2       3
```

```
Resultant Matrix C in Sparse Format:
i       j       v
3       3       3
0       2       3
1       1       4
2       0       3

Resultant Matrix C in Normal Format:
0 0 3
0 4 0
3 0 0
```

Add Two Polynomials



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

#define MAX_DEGREE 100

void addPolynomials(int poly1[], int poly2[], int result[], int degree1, int degree2) {
    int i;
    int maxDegree = (degree1 > degree2) ? degree1 : degree2;

    for (i = 0; i <= maxDegree; i++) {
        result[i] = ((i <= degree1) ? poly1[i] : 0) + ((i <= degree2) ? poly2[i] : 0);
    }
}

void printPolynomial(int poly[], int degree) {
    int i;
    for (i = degree; i >= 0; i--) {
        if (poly[i] != 0) {
            if (i < degree) {
                printf(" %c ", (poly[i] > 0) ? '+' : '-');
            }
            printf("%d", abs(poly[i]));
            if (i > 0) {
                printf("x");
                if (i > 1) {
                    printf("^%d", i);
                }
            }
        }
    }
    printf("\n");
}
```

```c
int main() {
    int degree1, degree2;
    int poly1[MAX_DEGREE + 1] = {0};
    int poly2[MAX_DEGREE + 1] = {0};
    int result[MAX_DEGREE + 1] = {0};

    printf("Enter the degree of the first polynomial: ");
    scanf("%d", &degree1);

    printf("Enter the coefficients of the first polynomial (from constant term to highest degree term):\n");
    for (int i = 0; i <= degree1; i++) {
        printf("Coefficient of x^%d: ", i);
        scanf("%d", &poly1[i]);
    }

    printf("Enter the degree of the second polynomial: ");
    scanf("%d", &degree2);

    printf("Enter the coefficients of the second polynomial (from constant term to highest degree term):\n");
    for (int i = 0; i <= degree2; i++) {
        printf("Coefficient of x^%d: ", i);
        scanf("%d", &poly2[i]);
    }

    addPolynomials(poly1, poly2, result, degree1, degree2);

    printf("First Polynomial: ");
    printPolynomial(poly1, degree1);

    printf("Second Polynomial: ");
    printPolynomial(poly2, degree2);

    printf("Sum of Polynomials: ");
    printPolynomial(result, (degree1 > degree2) ? degree1 : degree2);

    return 0;
}
```

```
KIIT0001@Utkarsh MINGW64 /d/Learning C 3rd Sem/assignments/assignment_aug9
$ ./add
Enter the degree of the first polynomial: 2
Enter the coefficients of the first polynomial (from constant term to highest degree term):
Coefficient of x^0: 1
Coefficient of x^1: 2
Coefficient of x^2: 3
Enter the degree of the second polynomial: 2
Enter the coefficients of the second polynomial (from constant term to highest degree term):
Coefficient of x^0: 1
Coefficient of x^1: 2
Coefficient of x^2: 3
First Polynomial: 3x^2 + 2x + 1
Second Polynomial: 3x^2 + 2x + 1
Sum of Polynomials: 6x^2 + 4x + 2
```