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

struct SparseElement
{
  int row;
  int col;
  int value;
};

struct SparseMatrix
{
  int rows;
  int cols;
  int numElements;
  struct SparseElement *data;
};

struct SparseMatrix *convertToSparse(int **matrix, int rows, int cols)
{
  int i, j;
  int numNonZero = 0;
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < cols; j++)
    {
      if (matrix[i][j] != 0)
      {
        numNonZero++;
      }
    }
  }

  struct SparseMatrix *sparseMatrix = (struct SparseMatrix
*)malloc(sizeof(struct SparseMatrix));
  sparseMatrix->rows = rows;
  sparseMatrix->cols = cols;
  sparseMatrix->numElements = numNonZero;
  sparseMatrix->data = (struct SparseElement *)malloc(numNonZero *
sizeof(struct SparseElement));

  int index = 0;
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < cols; j++)
    {
      if (matrix[i][j] != 0)
      {
        sparseMatrix->data[index].row = i;
        sparseMatrix->data[index].col = j;
        sparseMatrix->data[index].value = matrix[i][j];
        index++;
      }
    }
  }

  return sparseMatrix;
}

struct SparseMatrix *transposeSparse(struct SparseMatrix *matrix)
{
  int i, j;
  struct SparseMatrix *transposed = (struct SparseMatrix
*)malloc(sizeof(struct SparseMatrix));
  transposed->rows = matrix->cols;
  transposed->cols = matrix->rows;
  transposed->numElements = matrix->numElements;
  transposed->data = (struct SparseElement *)malloc(transposed-
>numElements * sizeof(struct SparseElement));

  int k = 0;

  for (i = 0; i < matrix->cols; i++)
  {
    for (j = 0; j < matrix->numElements; j++)
    {
      if (matrix->data[j].col == i)
      {
        transposed->data[k].row = matrix->data[j].col;
        transposed->data[k].col = matrix->data[j].row;
        transposed->data[k].value = matrix->data[j].value;
        k++;
      }
    }
  }

  return transposed;
}

void displaySparse(struct SparseMatrix *matrix)
{
  int i;
  printf("%d\t%d\t%d\n", matrix->rows, matrix->cols, matrix-
>numElements);
  for (i = 0; i < matrix->numElements; i++)
  {
```

```c
    printf("%d\t%d\t%d\n", matrix->data[i].row, matrix->data[i].col,
matrix->data[i].value);
  }
}

void displayMatrix(struct SparseMatrix *matrix)
{
  int i, j, k = 0;
  for (i = 0; i < matrix->rows; i++)
  {
    for (j = 0; j < matrix->cols; j++)
    {
      if (i == matrix->data[k].row && j == matrix->data[k].col)
      {
        printf("%d ", matrix->data[k++].value);
      }
      else
      {
        printf("0 ");
      }
    }
    printf("\n");
  }
}

int main()
{
  int rows, cols, i, j;
  printf("Enter number of rows: ");
  scanf("%d", &rows);
  printf("Enter number of columns: ");
  scanf("%d", &cols);

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

  printf("Enter the matrix:\n");
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < cols; j++)
    {
      scanf("%d", &matrix[i][j]);
    }
  }
  printf("\nThe matrix is\n");
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < cols; j++)
    {
      printf("%d ", matrix[i][j]);
    }
    printf("\n");
  }

  struct SparseMatrix *sparseMatrix = convertToSparse(matrix, rows,
cols);

  printf("\nSparse representation:\n");
  displaySparse(sparseMatrix);

  struct SparseMatrix *transposedMatrix = transposeSparse(sparseMatrix);
  printf("\nTransposed sparse representation:\n");
  displaySparse(transposedMatrix);
  printf("\nThe transposed matrix is:\n");
  displayMatrix(transposedMatrix);

  if (((3 * transposedMatrix->numElements) + 3) > (transposedMatrix->rows
* transposedMatrix->cols))
  {
    printf("\nTriple format is advantageous\n");
  }
  else
  {
    printf("\nTriple format is non-advantageous\n");
  }

  for (i = 0; i < rows; i++)
  {
    free(matrix[i]);
  }
  free(matrix);
  free(sparseMatrix->data);
  free(sparseMatrix);
  free(transposedMatrix->data);
  free(transposedMatrix);

  return 0;
}
```