CSE 3100 Systems Programming	Out: 09/21/16
Assignment #3	Due: 09/28/16

Exercise 1. ***Matrix Transposition (40 points)***

Fixed size matrices can be represented in C as two-dimensional arrays, i.e., arrays of arrays. For example we can declare a 3×3 *matrix of double-precision real values by*

```
double a[3][3];
```

However, if we want to pass `a` *as argument to a function, at a minimum we must specify the second dimension of the array in the function prototype as in*

```
print_matrix( double a[][3] )
{
  ...
}
```

This is needed by the compiler to compute where in memory `a`*'s elements reside. With this approach we would need another function to print, say, the transpose of a* 4×4 *matrix!*

Arrays of pointers allow us to implement functions that can be applied to matrices of arbitrary size, with actual matrix dimensions provided as function arguments. In this exercise you must implement several basic functions for working with matrices represented using pointers of arrays, including functions for

- *allocating/deallocating space for a matrix of given dimensions*
- *reading/printing a matrix from/to standard input/output*
- *building the transpose of a matrix*

Use these functions in conjunction with the provided main() function to read matrices from standard input and print their transpose to the standard output.

Exercise 2. ***Addition of sparse polynomials (60 points)***

Many of the polynomials arising in practice are sparse, i.e., they have a number of non-zero coefficients that is much smaller than the degree of the polynomial. In these cases using an array representation is wasteful, and polynomials are best represented as a linked list. In this exercise you are required to implement addition of sparse univariate polynomials with integer coefficients. Polynomials must be represented by your program as linked lists of monomials. The provided template code defines structures for representing monomials and polynomials and includes headers for several functions that operate on them. You are free to change the provided structures and headers as you see fit, as long as your program delivers the functionality described below.

Input: *Your program should read from the standard input 2 lines, each representing a polynomial. For each line the first integer represents the degree of the polynomial. This is followed by a number of pairs of integers that represent the coefficient and exponent of the monomials that comprise the polynomial. For example, the polynomial* $3 + 2x^4 + x^{10}$ *is represented by the sequence*

```
10 3 0 2 4 1 10
```

You may assume that monomials are given in increasing order of their exponents. *Coefficients are arbitrary non-zero integers; monomials with zero coefficients are always omitted.*

Output: *The program should print the result obtained by adding the two input polynomials using the same format as the one used for input. The output should never include zero coefficients; in case the result is the zero polynomial the output should be empty.*

Sample Input:	Sample Output:

`2 1 0 3 2`
`2 2 1 4 2`
	`2 1 0 2 1 7 2`