These terms are stored, in descending order of exponents, in positions terms[i][1], terms[i][2], ... Create the functions readPoly, printPoly, padd and pmult for this representation. Is this representation better or worse than the representation used in the text? (You may add declarations as necessary).

```
#define MAX_TERMS 101 /*maximum number of terms+1 */
#define MAX_POLYS 15 /*maximum number of polynomials */

typedef struct {
        float coef;
        int expon;
    } polynomial;
polynomial terms[MAX_POLYS][MAX_TERMS]
```

- The functions are implemented on my PC. I am not repeating the proofs of correctness of the functions because they are similar to the 1D version.
- Compared to the 1D version where we needed to restore the start and finish indexes of each polynomial separately, here we don't need to do that due to the fact that for each polynomial in row i, we are storing the no. of terms in terms[i][0].expon
- The 2D representation sacrifices space flexibility. If a polynomial has only 2 terms, you still reserve MAX_TERMS slots per row.