# 111-2 Data Structure

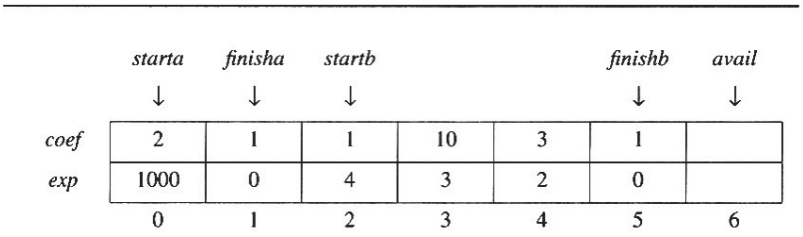
# Homework 2 Polynomials

## Question 1 (80%)

Suppose that we have two sparse polynomials A, B which are stored in the following representation in order to save space.

```
#define MAX_TERMS 100
/*size of terms array*/
typedef struct{
    float coef;
    int expon;
}polynomial;
polynomial terms [MAX_TERMS];
int avail = 0;
```

The index of the first term of A and B is given by startA and startB, respectively, and finishA and finishB give the index of the last term of A and B. The index of the next free location (to store the obtained result) in the array is given by avail (as shown below).



|  | *starta* ↓ | *finisha* ↓ | *startb* ↓ |  |  | *finishb* ↓ | *avail* ↓ |
|---|---|---|---|---|---|---|---|
| *coef* | 2 | 1 | 1 | 10 | 3 | 1 |  |
| *exp* | 1000 | 0 | 4 | 3 | 2 | 0 |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

**Figure 2.2:** Array representation of two polynomials

Please write a program to ***add*** the given two polynomials.

### Input format

The first line of a test case has two integers *m, n* representing the number of terms of polynomials A and B, respectively. From the second line, the first term represents coef[i] and the second term represents exp[i] (if for (i+2)th line) of the array representation in Figure 2.2.

## Constraints

*0 <= i <= 99*

*1 <= m <= 99, 1 <= n <= 99*

*-1000 <= exp[i] <= 1000, -1000 <= coef[i] <= 1000*

## Output format

Give the resulting polynomial arranged in descending order. For (i+1)th line, the first term represents coef[i] and the second term represents exp[i] of the array representation of the format in Figure 2.2.

## Sample input 1

```
2 4
2 1000
1 0
1 4
10 3
3 2
1 0
```

## Sample output 1

```
2 1000
1 4
10 3
3 2
2 0
```

## Sample input 2

```
3 1
3 10
5 4
1000 2
1 10
```

## Sample output 2

```
4 10
5 4
1000 2
```

## Question 2 (20%)

Continuing from Question 1, please write a program to ***multiply*** the given two polynomials. All the other rules are the same as in Question 1.

### Sample input 1

```
2 4
2 1000
1 0
1 4
10 3
3 2
1 0
```

### Sample output 1

```
2 1004
20 1003
6 1002
2 1000
1 4
10 3
3 2
1 0
```

### Sample input 2

```
3 1
3 10
5 4
1000 2
1 10
```

### Sample output 2

```
3 20
5 14
1000 12
```