**HW1**

**If your program takes more than a certain reasonable amount of time to finish, you get 0 points.**

**Every program should be compilable and runnable in cspro.**

**Every program should be written in C (at this time, No C++) and satisfy the requirements in the program specification.**

**Copy check will be done.**

1. Write a complete **C** program that reads a matrix in a **standard** matrix form of A=[aij], i=1,...,m and j=1,...,n from the “input.txt” file and obtains the transpose of the matrix and writes the output into the “output.txt” file in the **standard** matrix form. Your program should use **dynamically allocated arrays** and use the function with the header given as

**void transpose(int \*\*a, int \*\*b, int rows, int cols)**

where a is the rows x cols matrix to be transposed and b is the transposed matrix computed by transpose(). You can assume that all entries of the input matrix are integers. Note that the values of rows and cols may be **different** (The input matrix might not be square)**.**

**Ex.**

The example of the matrix in “input.txt”: (The first line contains the values of rows and cols of the input matrix and then rows x cols entries are given in the standard matrix form.)

**2 3**

**-1 0 3**

**2 4 0**

The matrix in “output.txt”:

**3 2**

**-1 2**

**0 4**

**3 0**

1. Write a complete **C** program that reads two polynomials in stdin and outputs the polynomial obtained in stdout by multiplying the two input polynomials. Your program should include and use the functions readPoly(), printPoly(), and pmult(). **Polynomials are represented by the method given in Figure 2.3 in your textbook (chapter 2 45page)**. The function readPoly() needs to prompt the user for the values of the coefficient and the exponent for each term in each polynomial. The function printPoly() writes the answer computed by pmult() to stdout by the standard polynomial form in x.

|  |  |
| --- | --- |
| **Ex.**(the first line corresponds to the number of the terms and the pairs of the coeff. and exp. follow. | |
| Input (stdin)  Input polynomial:  2[Enter]  3 1[Enter]  1 0[Enter]  Input polynomial:  2[Enter]  1 1[Enter]  -1 0[Enter] | Output (stdout)  3x^2–2x^1-1 |

1. Implement **Program 2.10: Sparse matrix multiplication (ch2 89page)** in your textbook by writing a complete C program. You should follow the same sparse matrix representation used in Program 2.10. Your program should include and use the function readMatrix() and printMatrix(). The function readMatrix() reads a matrix given in a **standard** matrix form of A=[aij], i=1,...,m and j=1,...,n into the matrix in the sparse matrix form. You need to call readMatrix() twice to obtain the input A and B matrices from “inputA.txt” and “inputB.txt”, respectively. The function printMatrix() prints out the result in the sparse matrix form obtained by the program 2.10 into a **sparse** matrix form into the file “output.txt”.

**Ex.**

InputA.txt (the format by the example of the problem 1):

2 3

4 0 5

0 1 0

InputB.txt:

3 2

1 0

0 1

0 -1

Output.txt (**the format by the sparse matrix form**):

2 2 3

0 0 4

0 1 -5

1 1 1

1. **Modify** **pmatch()** in your textbook (or lecture note)(**ch2 p168**) to a new function pmatch\_all() such that it prints out the starting index of every matched part of the input string and pattern in stdout. You need to write a complete C program that prompts the input in stdin:
   * No global variable is allowed.
   * A static allocation is possible.
   * If your algorithm does not use a modified pmatch(), you get 0 points.
   * Input:
     + first line: string  
       second line: pattern
     + Each input string contains at least one character and at most 30 characters.
   * Output:
     + Print out the starting index for a match at one line per match.
     + No output if there is no match.

|  |  |
| --- | --- |
| Ex. | |
| Input | Output |
| bbbbbabbbbbc  bbb | 0  1  2  6  7  8 |
| bbbbbabbbbbc  aa |  |