Problem 1 (70 pts) Implement the methods *minMult*, *maxMult*, and *printOrder*.

```
double minMult(int n, double [] d, double [][] M, int [][] P) {

   int i, j, k, diag;
   double min;

   for (i = 1; i <= n; i++)
      M[i][i] = 0.0;

   for(diag = 1; diag <= n-1; diag++)
      for (i = 1; i <= n - diag; i++)  {

         j = i + diag;
         M[i][j] = Double.MAX_VALUE;;

         for (k = i; k <= j - 1; k++)  {

            min = M[i][k] + M[k+1][j] + d[i-1]*d[k]*d[j];

            if (min < M[i][j]) {
               M[i][j] = min;
               P[i][j] = k;
            }
         }
      }
   return M[1][n];
}

double maxMult(int n, double [] d, double [][] M, int [][] P) {

   int i, j, k, diag;
   double max;

   for (i = 1; i <= n; i++)
      M[i][i] = 0.0;

   for(diag = 1; diag <= n-1; diag++)
      for (i = 1; i <= n - diag; i++)  {

         j = i + diag;
         M[i][j] = 0;

         for (k = i; k <= j - 1; k++)  {

            max = M[i][k] + M[k+1][j] + d[i-1]*d[k]*d[j];

            if (max < M[i][j]) {
               M[i][j] = max;
               P[i][j] = k;
            }
         }
      }
   return M[1][n];
}
```

```
void printOrder(int i, int j, int [][] P)  {

   int k = 0;
   if (i == j)
      System.out.print("A_" + i);

else {
   k = P[i][j];
   System.out.print("(");
   printOrder(i, k, P);
   printOrder(k+1, j, P);
   System.out.print(")");
}
}
```

Add one more example to the input file. Give the number of matrices *n*, and then the dimensions *d*. The goal is to maximize the ratio above. DO NOT WORK ON THIS PART TOGETHER. You may use any value of *n* up to 40, and the dimensions can be any number between 1 and 100. The one with the largest ratio will get **+5 extra credit points** on this assignment. If there are ties, the points will be divided. **WINNER:**

```
Max Cost: 1.82E7
Min cost: 2018.0
Largest ratio: 9018.830525272548

Using 39 matrices of dimensions:

A1:  1   x 100
A2:  100 x   1
A3:  1   x 100
A4:  100 x   1
A5:  1   x 100
.
.
.
A38: 100 x   1
A39: 1   x 100
```

Problem 2 (10 pts) You are almost done tracing through the MinMult function for the Chained Matrix Multiplication problem, with n = 6, and the dimensions of the 6 matrices as follows:

A_1	A_2	A_3	A_4	A_5	A_6
6×5	5×4	4×5	5×4	4×10	10×5

M	**1**	**2**	**3**	**4**	**5**	**6**
1	0	120	240	280	520	**600**
2		0	100	160	360	460
3			0	80	240	360
4				0	200	300
5					0	200
6						0

P	**1**	**2**	**3**	**4**	**5**	**6**
1		1	2	1	4	**2 or 4**
2			2	2	4	2
3				3	3	4
4					4	4
5						5
6						

$k = 1$: $(A_1) \times (A_2A_3A_4A_5A_6)$ requires $0 + 460 + (6 \times 5 \times 5) = 610$ multiplications.

$k = 2$: $(A_1A_2) \times (A_3A_4A_5A_6)$ requires $120 + 360 + (6 \times 4 \times 5) = 600$ multiplications.

$k = 3$: $(A_1A_2A_3) \times (A_4A_5A_6)$ requires $240 + 300 + (6 \times 5 \times 5) = 690$ multiplications.

$k = 4$: $(A_1A_2A_3A_4) \times (A_5A_6)$ requires $280 + 200 + (6 \times 4 \times 5) = 600$ multiplications.

$k = 5$: $(A_1A_2A_3A_4A_5) \times (A_6)$ requires $520 + 0 + (6 \times 10 \times 5) = 820$ multiplications.

M[1][6] = **600** P[1][6] = **2 OR 4** $A_1A_2A_3A_4A_5A_6$

The optimum order for multiplying matrices 1 – 6: $(A_1A_2)((A_3A_4)(A_5A_6))$ or $(A_1(A_2(A_3A_4)))(A_5A_6)$

The optimum order for multiplying matrices 1 – 5: $(A_1(A_2(A_3A_4))A_5$

The optimum order for multiplying matrices 2 – 6: $A_2((A_3A_4)(A_5A_6))$

Problem 3 (20 pts) Do the tracing for tabs *sa_2, sa_3, sa_4, and sa_5* the tracing in the "sequence_alignment_handout". For all of these, assume a mismatch penalty of 1 and a gap penalty of 2.

See file "SOLNS, sequence_alignment_handout.xlsx"