You should do the exercises in the "Floyd Handout" spreadsheet. I won't collect them, but you may be given such questions on an exam. 1 point penalty for each program that doesn't have your last name as the file and class name and does not have your name in the comment at the top. You must turn in your java files in a zipped folder.

Problem 1 Write the method *combin* which computes *n* choose *k*. It cannot use multiplication, division, or recursion. It must use $O(n)$ space (only 1-dimensional arrays are allowed). Write a (separate) program to test your method, and show the output. Use several test cases.

```
int combin(int n, int k)  {. . .}
```

FOR PROBLEMS 2 – 4, I have provided the template and the "main" method for you.

Problem 2 Implement Floyd's Algorithm and the printPath algorithm and test it on the graphs in files "graph11.text" and "graph22.text". **No credit if** you alter array *W* in `Floyd` or the array *P* in `printPath`.

```
void Floyd(int n, int [][] W, int [][] D, int [][] P)  {. . .}

void printPath(int s, int d, int [][] P)  {. . .}
```

Problem 3 Write a method that prints the optimum path (including endpoints) from vertex *s* to vertex *d*. Implement this and include this method in your program from Problem 2.

```
void printPathWithEndpoints(int s, int d, int [][] P)  {. . .}
```

Problem 4 Write the method below, which takes only *n* and the *P* matrix (passed back from Floyd's algorithm) and returns the largest number of edges in any single optimal cost path. Assume that the original graph was not missing any edges (i.e., no infinity signs).

```
int maxEdgesInOneOptPath(int n, int [][] P)  {
```