Topological Sort

Want to produce a linear ordering of the vertices in a DAG so that if there is an edge from vertex j to vertex k, then j appears before k in the linear ordering.

```
initialize visited[n] to false;
for each vertex v do                // makes sure we get every vertex in graph
   if (visted[ v ] == false)
      topsort(v);

void topsort(vertex v)
{
  visited[ v ] = true;

  for each neighbor w of v do
    if (visited[ w ] == false)
      topsort(w);

 output v;  // outputs the linear ordering in reverse order
}
```

Transitive Closure

Want to produce n x n matrix T such that T[i][j] = true if there exists a **path** from vertex i to vertex j, otherwse false.

```
bool** Warshall(bool** A, const int n)   // A is adjacency matrix
{
bool** T = new bool[n];

for (i = 0; i < n; i++)
  T[ i ] = new bool[n];

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
       T[ i ][ j ] = A[ i ][ j ];

   for (k = 0; k < n; k++)
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
        if (T[ i ][ j ] == false) T[ i ][ j ] = T[ i ][ k ] && T[ k ][ j ];

  return(T);
}
```