Graph - Transpose (A35)

Graph transpose G^T dari graph G didefinisikan sebagai graph G dengan seluruh edge-nya berbalik arah. Secara formal, jika ada edge u→ v pada G, maka ada edge v→ u pada G^T.

a. Implementasikan class Graph dengan menggunakan representasi adjacency matrix. Method-method public yang harus ada adalah:
   - `Graph(int N)` : constructor untuk membuat graph dengan N vertex
   - `void addEdge(int u, int v)` : untuk menambah edge u→v
   - `boolean edgeIsExist(int u, int v)`: untuk memeriksa apakah ada edge u → v
   - `Graph getTranspose()` : mengembalikan graph transpose dari G

   Anda boleh menambahkan method-method lain jika diperlukan.

b. Overide method `toString()` supaya dapat menampilkan adjacency matrix graph dengan format sebagai berikut:
   - String output terdiri dari *N* baris, masing-masing menandakan informasi ketetanggaan sebuah vertex pada graph.
   - Setiap baris terdiri dari *N* huruf yang dipisahkan spasi. Huruf '`T`' pada baris *u* kolom *v* menandakan ada edge dari vertex *u* ke vertex *v*. Huruf '`F`' menandakan tidak ada edge.

c. Untuk menguji program anda, tambahkan kelas Tester untuk menangani input/output seperti berikut ini:

   Spesifikasi Input

   Input diawali dengan sebuah bilangan bulat *N* ($1 \le N \le 1{,}000$) dan E ($1 \le E \le N^2$), yang menandakan jumlah vertex dan edge pada graph. *E* baris berikutnya masing-masing terdiri dari 2 bilangan bulat *u* dan *v* ($0 \le u,v \le N\text{-}1$), menandakan ada edge dari *u* ke *v*.

   Spesifikasi Output

   Output terdiri dari dua bagian yang dipisahkan dengan baris kosong. Bagian pertama menampilkan adjacency matrix graph pada input, sedangkan bagian kedua menampilkan adjacency matrix graph transpose-nya.

Contoh Input

```
4 7
0 1
2 1
3 2
1 3
2 0
0 0
1 2
```

Contoh Output

```
T T F F
F F T T
T T F F
F F T F

T F T F
T F T F
F T F T
F T F F
```