****

**BLG 372E**

**ANALYSIS OF ALGORITHMS II**

CRN: 22853

**REPORT OF HOMEWORK #3**

**Real Estate Matching**

Submission Date: 16.05.2014

**STUDENT NAME: TUĞRUL YATAĞAN**

**STUDENT NUMBER: 040100117**

**INSTRUCTOR: ZEHRA ÇATALTEPE**

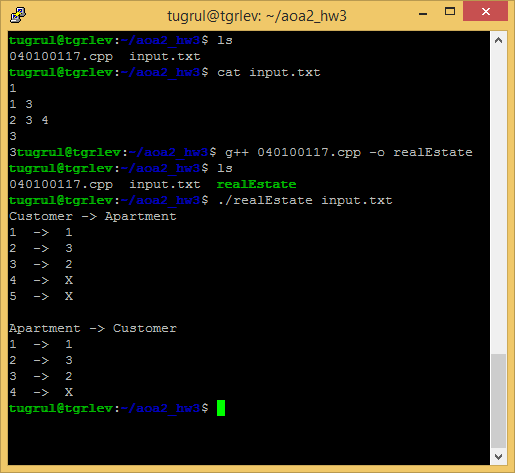
# **Building and Running**

The program built and compiled without any warning or error under g++ and the program executed with commands:

g++ 040100117.cpp –o realEstate

./realEstate input.txt

Sample output is below:



# **Data Structures and Variables**

Purpose of the all classes and methods explained in the source code as comment lines. In a nutshell;

* **BipartiteGraph** is the main class for algorithm. It can be used for all unary bipartite graph problems.
* Constructor of the **BipartiteGraph** class is allocates and initializes necessary data structures.
* **match** method of the **BipartiteGraph** class calls **augment** function for algorithm. **augment** is the main function of the algorithm.
* **match** method also prints the results as human readable format.
* **adjacencyMatrix** of the **BipartiteGraph** class is the matrix for keeping graph as two dimensional boolean array.

# **Analysis**

The Ford-Fulkerson’s Algorithm is used in this maximum bipartite matching problem. Ford-Fulkerson’s Algorithm is;

Main function:

FORDFULKERSON(G,E,s,t)

FOREACH e ∈ E

f(e) ← 0

Gf ← residual graph

WHILE (there exists augmenting path P)

f ← augment(f, P)

update Gf

ENDWHILE

RETURN f

Auxiliary function:

AUGMENT(f,P)

b ← bottleneck(P)

FOREACH e ∈ P

IF (e ∈ E)

f(e) ← f(e) + b

ELSE

f(eR) ← f(e) - b

RETURN f

(<http://www.cse.iitd.ernet.in/~Naveen/courses/CSL758/Ford%20Fulkerson%20Algorithm.ppt>)

When the capacities are integers, the runtime of Ford-Fulkerson is bounded by O(E \* f), where E is the number of edges in the graph and f is the maximum flow in the graph. This is because each augmenting path can be found in O(E) time and increases the flow by an integer amount which is at least 1. Therefore the time complexity becomes O(max\_flow \* E).

# **Graphs**

1. Initial graph



1. Source and sink are added



1. Algorithm starts







1. Algorithm finishes



1. Final graph, source and sink are removed

