BLG 372E

ANALYSIS OF ALGORITHMS II

CRN: 22853

REPORT OF HOMEWORK #1

Submission Date: 18.03.2014

STUDENT NAME: TUĞRUL YATAĞAN

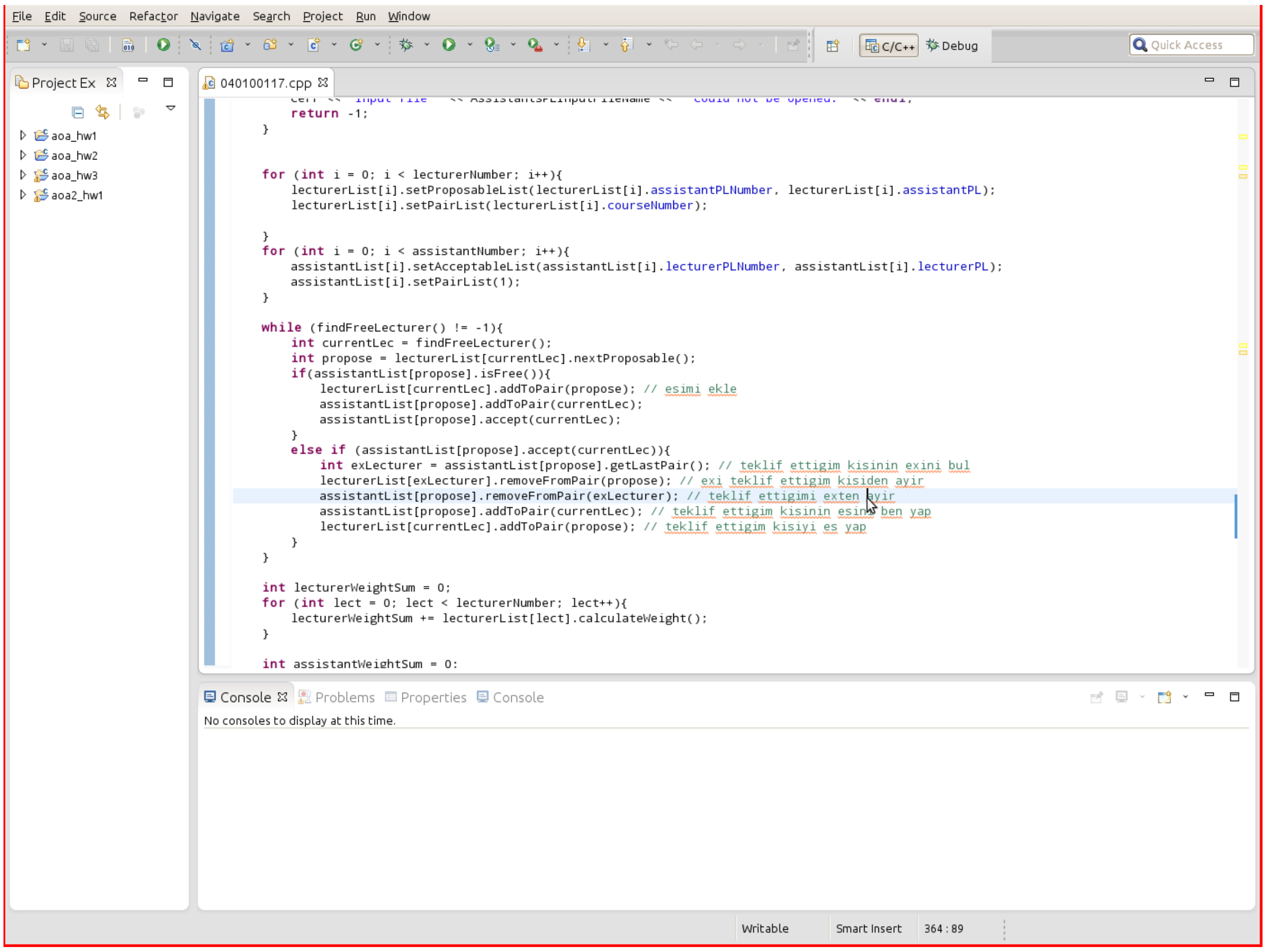
STUDENT NUMBER: 040100117

1. Introduction

In this project, we implement a lecturer-assistant assignment problem. Lecturers have preference lists for assistants. Assistants have preference list for the courses. One lecturer can have more than one course in the term. However an assistant can have only one course in each term.

2. Development and Operating Environments

Eclipse for C++ integrated development environment has been used to write the source code in Ubuntu 12.04 operation system and GNU g++ compiler has been used for compiling under Ubuntu 12.04 operation system.



```
        return -1;
    }


    for (int i = 0; i < lecturerNumber; i++){
        lecturerList[i].setProposableList(lecturerList[i].assistantPLNumber, lecturerList[i].assistantPL);
        lecturerList[i].setPairList(lecturerList[i].courseNumber);

    }
    for (int i = 0; i < assistantNumber; i++){
        assistantList[i].setAcceptableList(assistantList[i].lecturerPLNumber, assistantList[i].lecturerPL);
        assistantList[i].setPairList(1);
    }

    while (findFreeLecturer() != -1){
        int currentLec = findFreeLecturer();
        int propose = lecturerList[currentLec].nextProposable();
        if(assistantList[propose].isFree()){
            lecturerList[currentLec].addToPair(propose); // esimi ekle
            assistantList[propose].addToPair(currentLec);
            assistantList[propose].accept(currentLec);
        }
        else if (assistantList[propose].accept(currentLec)){
            int exLecturer = assistantList[propose].getLastPair(); // teklif ettigim kisinin exini bul
            lecturerList[exLecturer].removeFromPair(propose); // exi teklif ettigim kisiden ayir
            assistantList[propose].removeFromPair(exLecturer); // teklif ettigimi exten ayir
            assistantList[propose].addToPair(currentLec); // teklif ettigim kisinin esini ben yap
            lecturerList[currentLec].addToPair(propose); // teklif ettigim kisiyi es yap
        }
    }

    int lecturerWeightSum = 0;
    for (int lect = 0; lect < lecturerNumber; lect++){
        lecturerWeightSum += lecturerList[lect].calculateWeight();
    }

    int assistantWeightSum = 0;
```

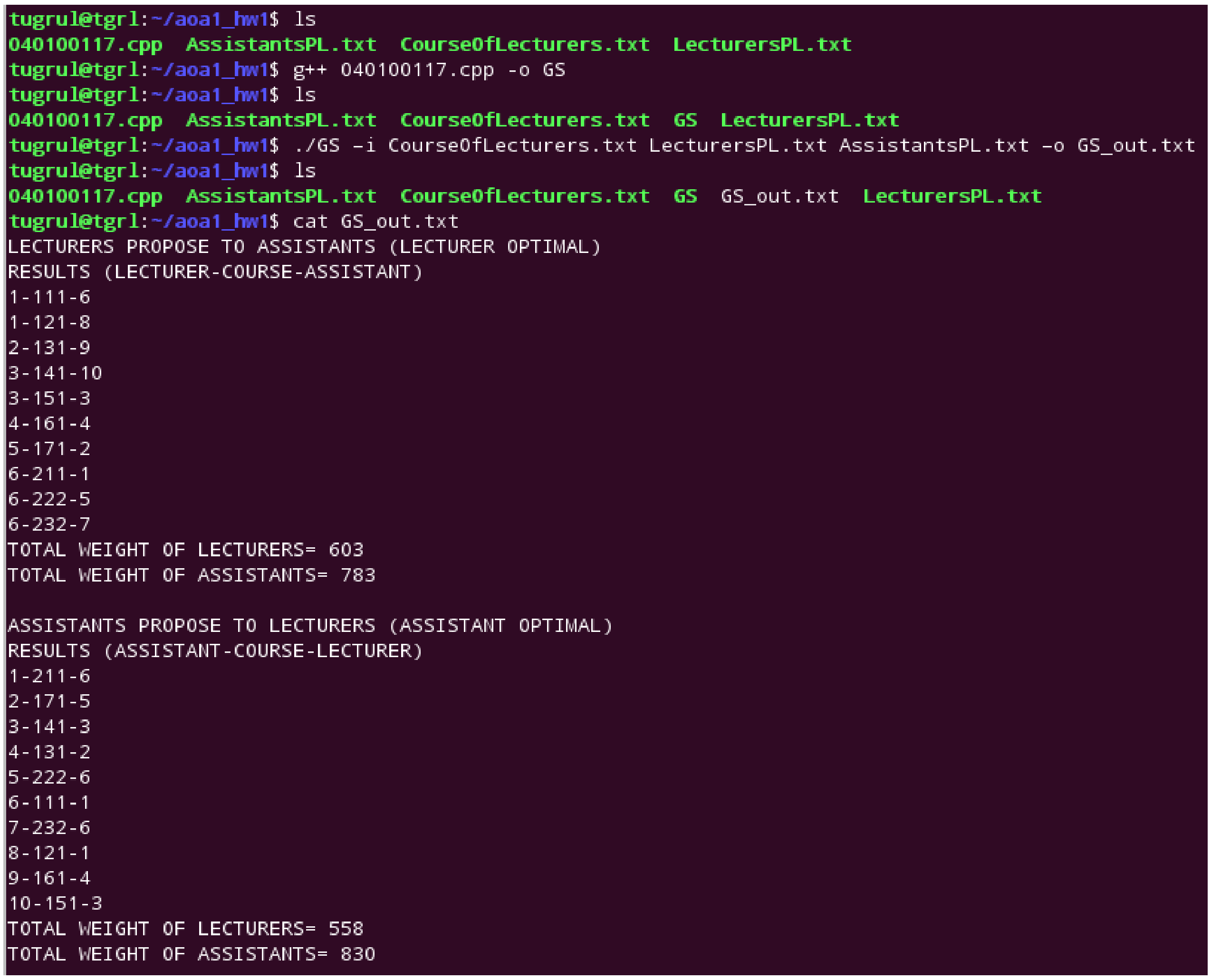
No consoles to display at this time.

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

```
g++ 040100117.cpp -o GS

./GS -i CourseOfLecturers.txt LecturersPL.txt AssistantsPL.txt  -o GS_out.txt
```

Sample output is below:



```
tugrul@tgrl:~/aoa1_hw1$ ls
040100117.cpp  AssistantsPL.txt  CourseOfLecturers.txt  LecturersPL.txt
tugrul@tgrl:~/aoa1_hw1$ g++ 040100117.cpp -o GS
tugrul@tgrl:~/aoa1_hw1$ ls
040100117.cpp  AssistantsPL.txt  CourseOfLecturers.txt  GS  LecturersPL.txt
tugrul@tgrl:~/aoa1_hw1$ ./GS -i CourseOfLecturers.txt LecturersPL.txt AssistantsPL.txt -o GS_out.txt
tugrul@tgrl:~/aoa1_hw1$ ls
040100117.cpp  AssistantsPL.txt  CourseOfLecturers.txt  GS  GS_out.txt  LecturersPL.txt
tugrul@tgrl:~/aoa1_hw1$ cat GS_out.txt
LECTURERS PROPOSE TO ASSISTANTS (LECTURER OPTIMAL)
RESULTS (LECTURER-COURSE-ASSISTANT)
1-111-6
1-121-8
2-131-9
3-141-10
3-151-3
4-161-4
5-171-2
6-211-1
6-222-5
6-232-7
TOTAL WEIGHT OF LECTURERS= 603
TOTAL WEIGHT OF ASSISTANTS= 783

ASSISTANTS PROPOSE TO LECTURERS (ASSISTANT OPTIMAL)
RESULTS (ASSISTANT-COURSE-LECTURER)
1-211-6
2-171-5
3-141-3
4-131-2
5-222-6
6-111-1
7-232-6
8-121-1
9-161-4
10-151-3
TOTAL WEIGHT OF LECTURERS= 558
TOTAL WEIGHT OF ASSISTANTS= 830
```

3. Data Structures and Variables

A main Matching class is inherited from both Lecturer and Assistant class for Gale-Shapley algorithm. Lecturer and Assistant classes has specific attributes and variables for their purposes.

```
class Matching{ // base class for propose-reject algorithm

        int *proposable; // list of member who candidate for propose

        int *acceptable; // list of member who candidate for acceptance

        int currentPairNumber; // current number of pair

        int maxProposableNumber; // maximum number of propose list

        int maxPairNumber; // maximum number of pair list

protected:

        int maxAcceptanceNumber; // maximum number of acceptance list
```

```
public:
	int *pair; // pair list
	void setProposableList(int, int *); // initialization for proposable list
	void setPairList(int); // initialization for pair list
	void setAcceptableList(int, int *); // initialization for acceptance list
	int nextProposable(); // calculates next element suitable for propose
	void addToPair(int); // adds to pair list
	void removeFromPair(int); // removes pair list
	int getLastPair(); // returns pair
	int multiGetLastPair(); // return last pair for list
	bool accept(int); // checks if propose is acceptable
	bool multiAccept(int); // checks if propose is acceptable for list
	bool isFull(); // is pair list full
	bool isFree(); // is pair list empty
	Matching(); // default constructor
};

class Lecturer: public Matching { // lecturer class inherited from matching
public:
	int courseNumber;
	int assistantPLNumber;
	int *courses; // course list
	int *assistantPL; // assistant preference list
	void addCourses(int, int *); // initialization for course list
	void addAssistantPL(int, int *); // initialization for assistant pref. list
	bool searchCourse(int); // searches course
	int calculateWeight(); // calculates weight for lecturer
};
class Assistant: public Matching {
public:
	int coursePLNumber;
	int lecturerPLNumber;
	int *coursePL; // course preference list
	int *lecturerPL; // lecturer preference list
```

```
        void addCoursePL(int, int *); // initialization for course list
        void addLecturerPL(int, int *); // initialization for lecturer list
        int findInLecturerPL(int); // searches lecturer
        int calculateWeight(); // calculates weight for lecturer
};
```

4. Analysis

Main matching algorithm is:

```
while (findFreeLecturer() != -1){ // bosta hoca var mı
        int currentLec = findFreeLecturer(); // bos hoca ata
        int propose = lecturerList[currentLec].nextProposable();
        // teklif edilebilecek kisi bul
        if(assistantList[propose].isFree()){ // teklif ettigim bos mu
                lecturerList[currentLec].addToPair(propose); // esi ekle
                assistantList[propose].addToPair(currentLec); // esimi bana ekle
                assistantList[propose].accept(currentLec); // esim beni kabul etsin
        }
        else if (assistantList[propose].accept(currentLec)){
        // teklif ettigim beni tercih ediyor mu
                int exLecturer = assistantList[propose].getLastPair();
                // teklif ettigim kisinin eski esini bul
                lecturerList[exLecturer].removeFromPair(propose);
                // eski esi teklif ettigim kisiden ayir
                assistantList[propose].removeFromPair(exLecturer);
                // teklif ettigimi eski sevgiliden ayir
                assistantList[propose].addToPair(currentLec);
                // teklif ettigim kisinin esini ben yap
                lecturerList[currentLec].addToPair(propose);
                // teklif ettigim kisiyi benim esim yap
        }
}
```

Algorithm above is implementation of this propose and reject algorithm pseudo code:

```
while ∃ free man m who still has a woman w to propose to {
       w = m's highest ranked woman to whom he has not yet proposed to
       if w is free
         (m, w) become engaged
       else some pair (m', w) already exists
         if w prefers m to m'
           (m, w) become engaged
           m' becomes free
    }
}
```

Complexity of the algorithm mainly relates on while loop. So length of the maximum preference list determinative for complexity.

Length of the preference list for assistants and lecturers is = n

Complexity is $O(n^2)$

5. Conclusion

During this homework, I have become more familiar with the concept of matching algorithms and analysis of the algorithms. I had the chance to intensify my knowledge about instructing good and efficient algorithms.

Illustration of matching algorithm with tables:

	Lecturers Assistant Preference List											Course Number
1)	10	8	6	4	3	2	1	9	7	5		2
2)	7	1	3	6	10	5	9	2	4	8		1
3)	10	7	6	4	5	1	3	8	9	2		2
4)	10	7	6	4	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	1	4	6	3	3	6	1	6	5
5)	6	6	3	4	2	5	1	3	6	1
6)	1	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	2	3	6	4	3	6
8)	5	6	1	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	3	5	4	1	2	1	3

	Lecturers Assistant Preference List											C.N.
1)	10	8	6	4	3	2	1	9	7	5		2
2)	7	1	3	6	10	5	9	2	4	8		1
3)	10	7	6	4	5	1	3	8	9	2		2
4)	10	7	6	4	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	1	4	6	3	3	6	1	6	5
5)	6	6	3	4	2	5	1	3	6	1
6)	1	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	2	3	6	4	3	6
8)	5	6	1	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	3	5	4	1	2	1	3

	Lecturers Assistant Preference List											C.N.
1)	**10**	**8**	6	4	3	2	1	9	7	5		2
2)	**7**	1	3	6	10	5	9	2	4	8		1
3)	10	7	6	4	5	1	3	8	9	2		2
4)	10	7	6	4	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	1	4	6	3	3	6	1	6	5
5)	6	6	3	4	2	5	1	3	6	1
6)	1	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	**2**	3	6	4	3	6
8)	5	6	**1**	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	3	5	4	**1**	2	1	3

	Lecturers Assistant Preference List											C.N.
1)	**10**	**8**	6	4	3	2	1	9	7	5		2
2)	**7**	1	3	6	10	5	9	2	4	8		1
3)	**10**	7	6	4	5	1	3	8	9	2		2
4)	10	7	6	4	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	1	4	6	3	3	6	1	6	5
5)	6	6	3	4	2	5	1	3	6	1
6)	1	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	**2**	3	6	4	3	6
8)	5	6	**1**	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	**3**	5	4	**1**	2	1	3

	Lecturers Assistant Preference List											C.N.
1)	**10**	**8**	**6**	4	3	2	1	9	7	5		2
2)	**7**	1	3	6	10	5	9	2	4	8		1
3)	**10**	7	6	4	5	1	3	8	9	2		2
4)	10	7	6	4	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	1	4	6	3	3	6	1	6	5
5)	6	6	3	4	2	5	1	3	6	1
6)	**1**	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	**2**	3	6	4	3	6
8)	5	6	**1**	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	**3**	5	4	**1**	2	1	3

	Lecturers Assistant Preference List											C.N.
1)	**10**	**8**	**6**	4	3	2	1	9	7	5		2
2)	**7**	1	3	6	10	5	9	2	4	8		1
3)	**10**	7	6	**4**	5	1	3	8	9	2		2
4)	10	7	6	4	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	1	4	6	**3**	3	6	1	6	5
5)	6	6	3	4	2	5	1	3	6	1
6)	**1**	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	**2**	**3**	6	4	3	6
8)	5	6	**1**	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	**3**	5	4	**1**	2	1	3

	Lecturers Assistant Preference List											C.N.
1)	**10**	**8**	**6**	4	3	2	1	9	7	5		2
2)	**7**	1	3	6	10	5	9	2	4	8		1
3)	**10**	7	6	**4**	5	1	3	8	9	2		2
4)	10	7	6	**4**	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	**1**	4	6	**3**	3	6	1	6	5
5)	6	6	3	4	2	5	1	3	6	1
6)	**1**	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	**2**	**3**	6	4	3	6
8)	5	6	**1**	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	**3**	5	4	**1**	2	1	3

	Lecturers Assistant Preference List											C.N.
1)	**10**	**8**	**6**	4	3	2	1	9	7	5		2
2)	**7**	1	3	6	10	5	9	2	4	8		1
3)	**10**	7	6	**4**	**5**	1	3	8	9	2		2
4)	10	7	6	**4**	1	5	9	8	3	2		1
5)	7	3	2	4	5	9	8	1	6	10		1
6)	1	4	5	7	10	8	9	3	2	6		3

	Assistants Lecturer Preference List									
1)	6	4	3	5	3	6	2	6	1	1
2)	5	6	3	3	2	1	6	1	4	6
3)	3	5	6	6	4	1	1	6	2	3
4)	2	**1**	4	6	**3**	3	6	1	6	5
5)	6	6	**3**	4	2	5	1	3	6	1
6)	**1**	6	5	1	3	2	4	6	6	3
7)	1	6	1	5	**2**	**3**	6	4	3	6
8)	5	6	**1**	1	3	2	4	6	3	6
9)	6	3	6	5	4	2	6	3	1	1
10)	6	6	6	**3**	5	4	**1**	2	1	3

Final condition is:

	Lecturers Assistant Preference List											C.N.	
1)	**10**	**8**	**6**	4	3	2	1	9	7	5		2	81+64
2)	**7**	**1**	3	6	10	5	**9**	2	4	8		1	16
3)	**10**	**7**	6	4	5	**1**	**3**	8	9	2		2	100+16
4)	**10**	7	6	**4**	1	5	9	8	3	2		1	49
5)	**7**	**3**	**2**	4	5	9	8	1	6	10		1	64
6)	**1**	4	**5**	**7**	10	8	9	3	2	6		3	100+64+49
													603

	Assistants Lecturer Preference List										
1)	**6**	4	3	5	3	6	2	6	1	1	100
2)	**5**	6	3	3	2	1	6	1	4	6	100
3)	**3**	5	6	6	4	1	1	6	2	3	100
4)	2	1	**4**	6	3	3	6	1	6	5	64
5)	**6**	6	3	4	2	5	1	3	6	1	100
6)	**1**	6	5	1	3	2	4	6	6	3	100
7)	1	**6**	1	5	2	3	6	4	3	6	81
8)	5	6	**1**	1	3	2	4	6	3	6	64
9)	6	3	6	5	4	**2**	6	3	1	1	25
10)	6	6	6	3	5	4	**1**	2	1	3	16
											783