Class Project Protocols

Project Policy

- Code must be original, and may not be copied or shared from any other source, except as provided by the class instructor
- Plagiarism will be very strictly punished

Project Submissions

- Each project has associated with it a *deadline due date*.
- For each project, you should submit a zip file containing the following:
  - All source files
  - `README.txt`, containing documentation, analysis and other comments. (This could a plaintext or Word file)
  - You may read the input from a database file or excel file and show it in GUI directly.
  - If you have any, sample run outputs should be able to export it to an excel file and show it on the GUI as well.

Project Grading

Projects will be graded based on the following criteria:

- *Correctness* -- the program system works **according to project requirements**
  - Source code
  - no syntax errors, no run-time errors, and the correct output must be produced
  - improper input gracefully handled, allocated space returned after it no longer is needed, and error-checking performed after calling any system or library function
- *Testing, timing* -- correct implementation of all functions needed for the complete project but not part of the project specifications
  - code needed for I/O
  - code needed for testing
  - code needed for timing
- *Program structure* -- well modularized, no gross inefficiencies (space or time)
- *Internal documentation* -- concise comments, indentation, descriptive variable names, explain "magic" numbers
  - `README.txt/doc`
  - how to use the program
  - description of data structure implementation (some of this may be internal)
  - theoretical time analysis of the underlying algorithms
  - production of a function (based on your timing and analysis) to predict the time required by your system for larger instances of the problem
  - accurate analysis of the space requirements

Algorithms Project (3rd Year)

Efficient School Timetable/Exams

***** Due SUNDAY JAN 3 2011 *****

The school timetable problem that we are intended to solve in this project can be described as follows:

In a department like computer engineering or a school of engineering in general, we suffer every semester from setting a suitable a timetable schedule for professors, classrooms, courses, and students as well as scheduling exams. The timetable/exam scheduling usually takes long time to be stabilized. The reason behind that is the hard and soft constraints that need to be taken into consideration during forming the timetable/exam scheduling. Although too much effort and time spent in scheduling, some of the professors as well as students complain about their schedule by the end of the day. Therefore, we want to make sure that our timetable/exams for the department/school is the best we could get and convince everybody by that. One way to do so is to automate the timetable generation process and just balm the computer for any inconvenience ☺. Therefore, this is your job in this class. The following are description of what you need to do.

In this project, we will study some of the important techniques and algorithms that are widely used in practice such as coloring algorithms and graph theory. Also, the intention behind this project is to design efficient algorithms and efficient software to solve hard problems. So, it is your responsibility to handle each and every issue related to the project efficiency in terms of design and software development. In addition to the timetable schedule, we want to utilize the similar components used for the timetable in scheduling exams as well. That enables 2X1 product.

In addition to the efficiency of the used algorithms, a simple user interface is a must. Therefore, the following figures are samples to the required user interface.

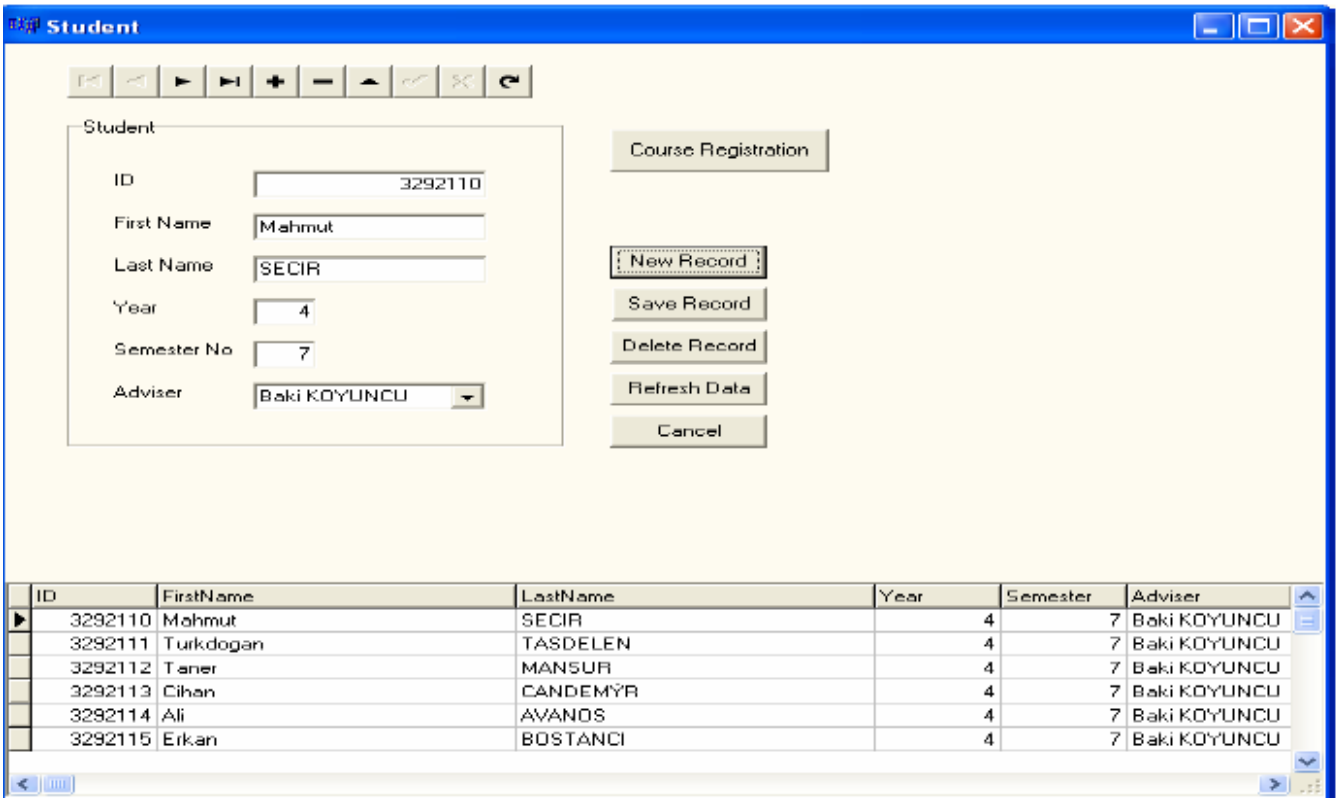


Figure 1: Student Data Entrance

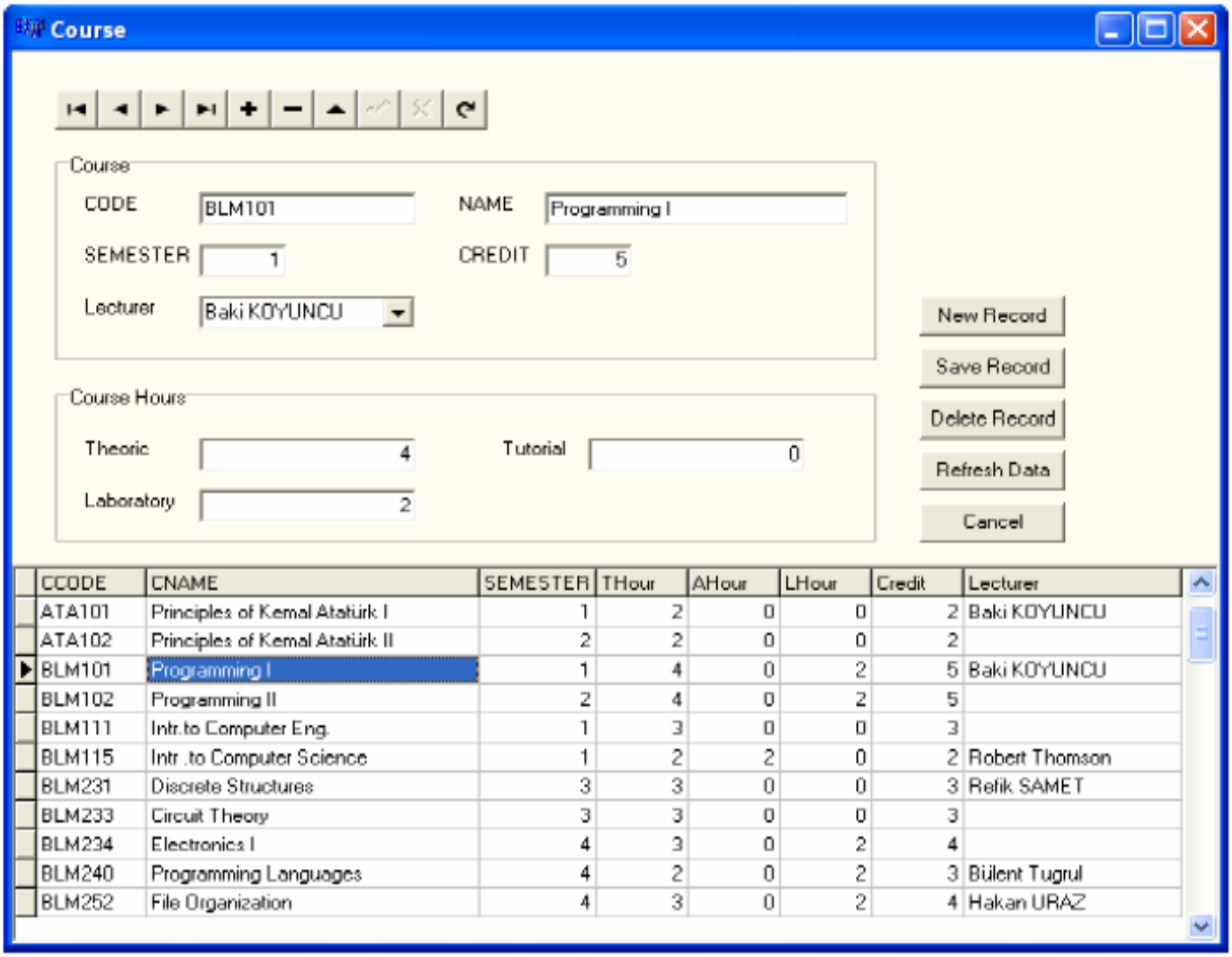


Figure 2 : Course Data Entrance

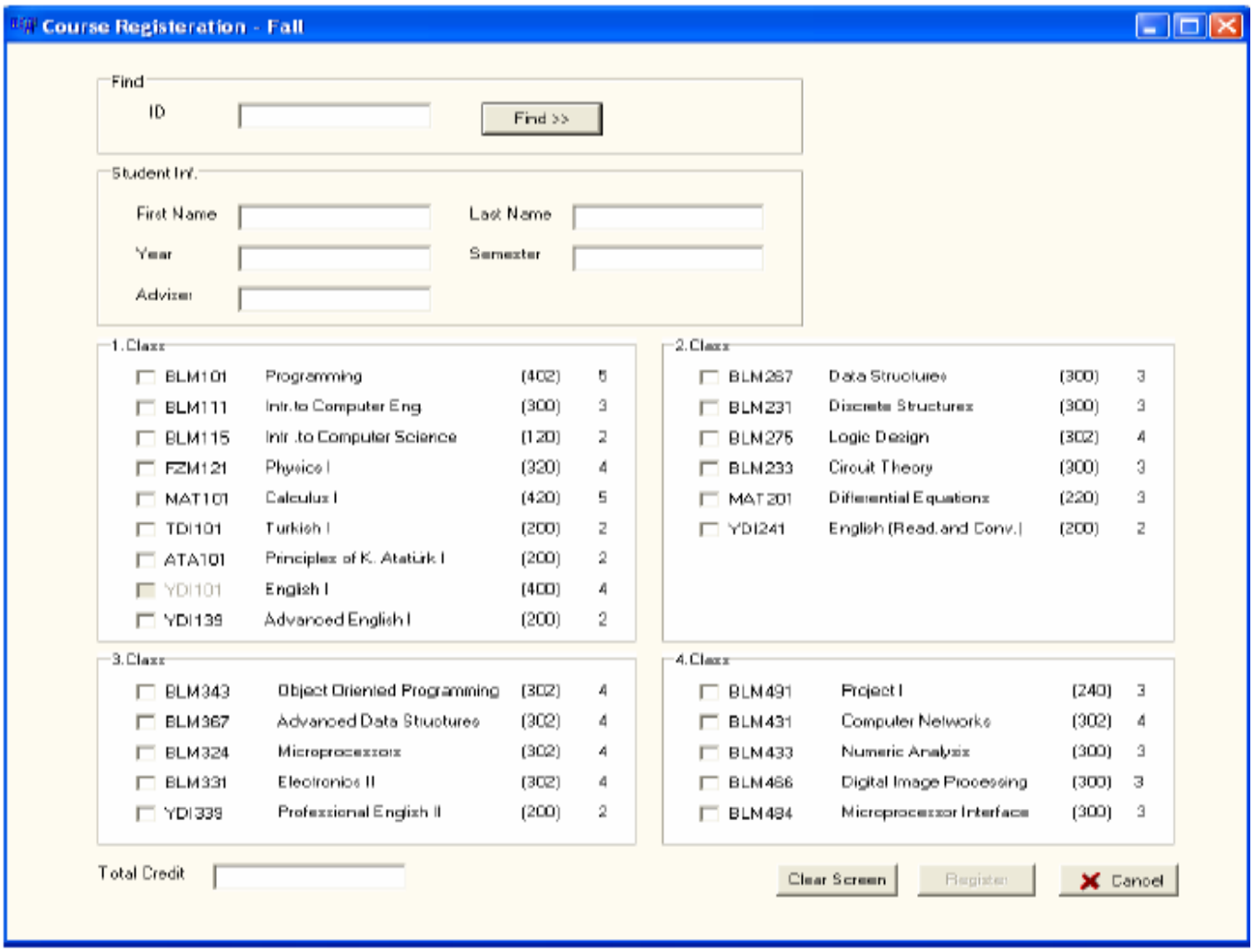


Figure 3: Course Registration Entrance

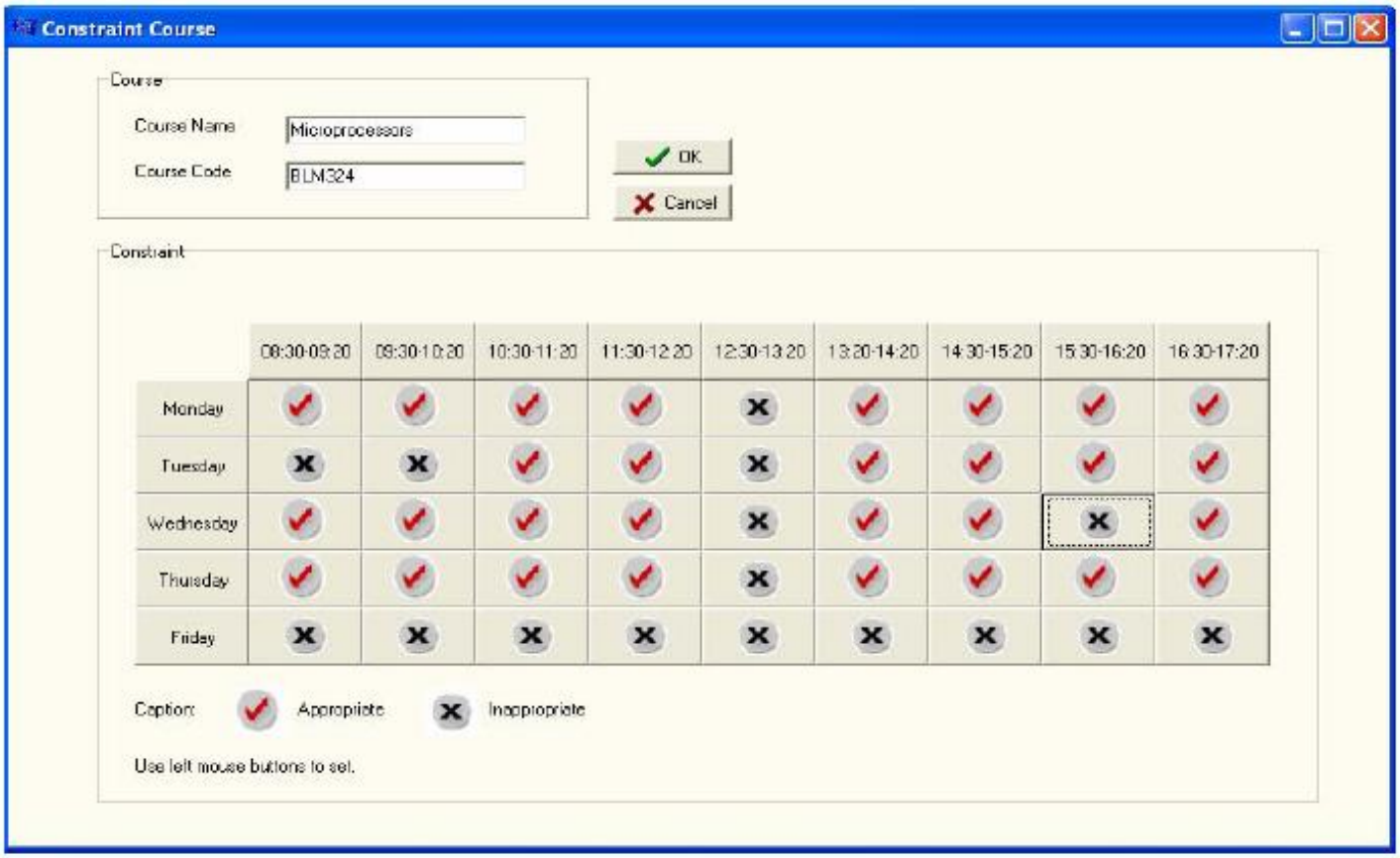
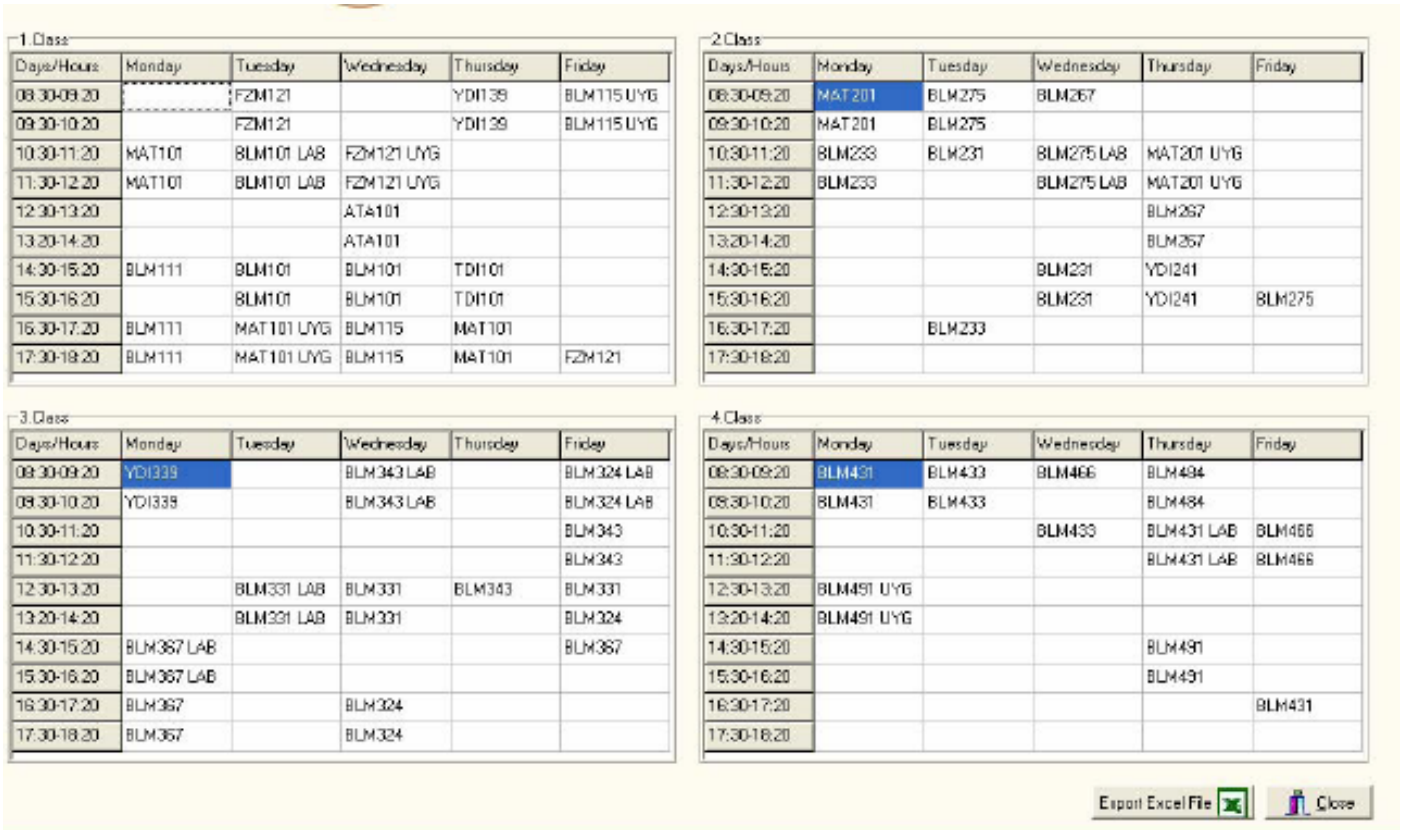


Figure 4 : Constraint Set



1.Class

Days/Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:30-09:20		FZM121		YDI139	BLM115 UYG
09:30-10:20		FZM121		YDI139	BLM115 UYG
10:30-11:20	MAT101	BLM101 LAB	FZM121 UYG		
11:30-12:20	MAT101	BLM101 LAB	FZM121 UYG		
12:30-13:20			ATA101		
13:20-14:20			ATA101		
14:30-15:20	BLM111	BLM101	BLM101	TDI101	
15:30-16:20		BLM101	BLM101	TDI101	
16:30-17:20	BLM111	MAT101 UYG	BLM115	MAT101	
17:30-18:20	BLM111	MAT101 UYG	BLM115	MAT101	FZM121

2.Class

Days/Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:30-09:20	MAT201	BLM275	BLM267		
09:30-10:20	MAT201	BLM275			
10:30-11:20	BLM233	BLM231	BLM275 LAB	MAT201 UYG	
11:30-12:20	BLM233		BLM275 LAB	MAT201 UYG	
12:30-13:20				BLM267	
13:20-14:20				BLM267	
14:30-15:20			BLM231	YDI241	
15:30-16:20			BLM231	YDI241	BLM275
16:30-17:20		BLM233			
17:30-18:20					

3.Class

Days/Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:30-09:20	YDI339		BLM343 LAB		BLM324 LAB
09:30-10:20	YDI339		BLM343 LAB		BLM324 LAB
10:30-11:20					BLM343
11:30-12:20					BLM343
12:30-13:20		BLM331 LAB	BLM331	BLM343	BLM331
13:20-14:20		BLM331 LAB	BLM331		BLM324
14:30-15:20	BLM367 LAB				BLM367
15:30-16:20	BLM367 LAB				
16:30-17:20	BLM367		BLM324		
17:30-18:20	BLM367		BLM324		

4.Class

Days/Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:30-09:20	BLM431	BLM433	BLM466	BLM484	
09:30-10:20	BLM431	BLM433		BLM484	
10:30-11:20			BLM433	BLM431 LAB	BLM466
11:30-12:20				BLM431 LAB	BLM466
12:30-13:20	BLM491 UYG				
13:20-14:20	BLM491 UYG				
14:30-15:20				BLM491	
15:30-16:20				BLM491	
16:30-17:20					BLM431
17:30-18:20					

Export Excel File | Close

Figure 5: Resultant Timetable/Exams

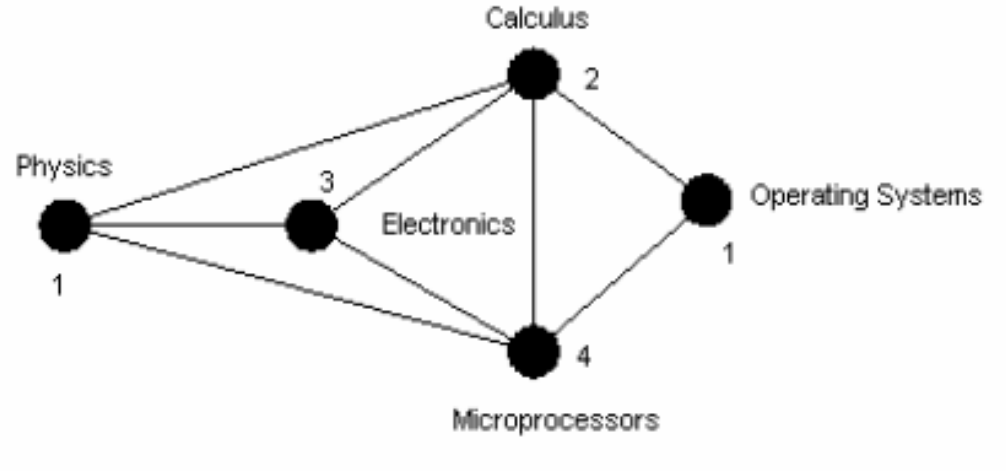


Figure q6: Sample graph to the courses and their relation

Table 1: Conflict table

	Physics	Calculus	Electronics	Microprocessors	Operating Systems
Physics		*	*	*	
Calculus	*		*	*	*
Electronics	*	*		*	
Microprocessors	*	*	*		*
Operating Systems		*		*	

The cause for potential conflict could be any of the following example restrictions:

- Courses Calculus and Electronics might be taught by the same professor,
- Courses Microprocessors and Operating Systems might be taken by the same student.

- **Hard Constraints** (constraints that cannot be neglected and must be satisfied)
  1- A teacher can only teach in a single place at a time.
  2- A teacher can only give one lecture at a time.
  3- A room can only host one lecture at a time.
  4- A student can only attend one lecture at a time.
  5- No more than a teacher is scheduled to teach in a room each time.
  6- Every teacher must have scheduled all his/her hours.
  7- Every student must have scheduled all his/her hours.

- **Soft Constraints :** (constraints over that we can pass sometimes but it is preferable not to do so and every time we violate them this fact is reflected on the overall efficiency of the timetable)
  1. A teacher should not teach more than 6 hours a day.
  2. A student should not have more than 8 hours a day.
  3. There shouldn't be gaps in the activity of the teachers.

4. There shouldn't be gaps in the activity of the students.

Please note that there is a difference between Labs, tutorials, and Lectures. Also, TAs are assigned only to Labs and tutorials.

The same rules are applied for exams when it is applicable; however, a student cannot have more than two exams per day. In addition, hard constraint #5 is relaxed for the exam schedule.

Guidelines Articles:

- **A Survey of Automated Timetabling**
- **Graph Coloring Conditions for the Existence of Solutions to the Timetable Problem**
- **A University Timetabling System based on Graph Colouring and Constraint Manipulation** **(Very important)**
- **http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.95.3879**
- **http://webdocs.cs.ualberta.ca/~joe/Coloring/#Graph.Colorers**
- **http://www.adaptivebox.net/CILib/code/gcpcodes_link.html**
- **http://shah.freeshell.org/graphcoloring/**
- **http://www.emo.org.tr/ekler/76e76856c7fea3b_ek.pdf**

- **Deliverables:**
  1- Survey on different solutions to the problem .

  2- Justification to data structures and algorithms used . In other words , what data structures you used and why?

  3- Source code of your implementation.

  4- Readme file.

5- Some test cases of your design.

6- Hard and soft-copy report describing the used methods

- **Project Grading (100/105)**

1) The project starts by designing a data structure to enter the courses and instructor requirements. A simple and efficient data structure and user interface is required. (8%)

2) A GUI and efficient data structure for the student registration are required. (8%)

3) GUI for setting the constraints (5%)

4) Visualizing the relation between the courses (9%)

5) Generating the conflict matrix (table) (10%)

6) Choosing and implementing the best coloring algorithm (20%)

7) Satisfying all hard and soft constraints (15%)

8) Generating the final tables (5%)

9) Considering the Exam schedule using the same coloring algorithm (15%)

10) Efficient programming (comments, documentation, structure, source code …, etc.) (5%)

11) Final Report (table of contents , Survey on the problem solutions , clear description to the data structure and algorithms used, test cases , performance measures to the algorithms and the running time of the software) (5%).