PROJECT REPORT

CS101 Project 2015

Time-Table Scheduling and Optimisation

Group CUSE

R.Basuhi, 140010037

Yashaswini K Murthy, 140010054

Shachi Shailesh Deshpande, 140110047

Table of Contents

1 Introduction

The following document describes the functional and non-functional requirements for the Time-Table Scheduling and Optimisation program to be submitted as a Software Project as part of our CS101 course-Spring Semester 2015. The features expressed in this Software Requirements Specification document are intended to be implemented to the maximum possible extent in our project. The system will be developed in such a way to provide easy addition of enhanced features, which may be desired in subsequent versions, if any.

Definitions, Acronyms and Abbreviations used,

- **Time-Table Scheduling:** In the context of our project, Time-Table scheduling is a constraint based problem wherein one inputs the constraints pertaining to a universities' course subjects, professor's availability and time-slots.
- **Soft Constraints:** Constraints that are negotiable to a certain extent for achieving the optimal solution.
- **Hard Constraints:** If the problem mandates that the constraints be satisfied, they are called hard constraints.
- **Perturbation:** In our context, the perturbation method means we take a feasible solution and make random changes (called perturbations) and try to find the most optimum timetable among the ones generated.
- **TAs:** Teaching Assistants handle Labs and Tutorials that supplement the lectures in the predeterimined course structure that we have created

Freshman Year Course Structure:

Theory Courses:

Course Code	Lecture Hours per week	Tutorial Hours per week
PH107	2	1
CH107	3	0
MA106	2	1
IC102	3	0
BB101	3	0

Lab Courses:

PH117L	
CH117L	

Note:

- The entire Freshmen batch is sub-divided into 4 batches, namely D1, D2, D3 and D4. D1 and D2, out of these have morning labs and afternoon lectures, whereas the vice-versa is true for D3 and D4.
- If you are a professor or a TA, you'll be instructing 2 batches for the subject you choose over the semester.
- Each tutorial or lecture is 1 hour long in duration and each lab is 4 hours long.

2 Problem Statement

Given, input by professors and TAs as to their preferred days and slots for teaching a particular subject, generate an optimised Time-Table subject to hard and soft constraints.

3 Requirements

The computer on which the program is to be run must have software supportability for C++ programs. Codeblocks with the GNU GCC Compiler must be installed. No other toolchain, packages, hardware requirements are needed.

4 Testing Data and Strategy

4.1 Test Data

1. If the teaching requirements of a particular subject are fulfilled, then, the next Professor who demands that subject will be intimated to change his subject or exit the form. Input through these forms will not cease until all the courses have a professors and TAs.

2. Slot clashes have to be avoided at all costs. If, say, all the professors give high preference for a particular day, maybe Monday, then the optimisation will be very low and perturbations won't be effective in generating such a time-table. Many will have to compromise on their preferences.

3. If there is a preference clash between TAs and Professors, Professors are given more weightage.

4.2 Testing Methodology:

Default cases will be generated by us to check and remove logical errors, if any. The default cases will be selected such that they cover both generic as well as extreme scenarios

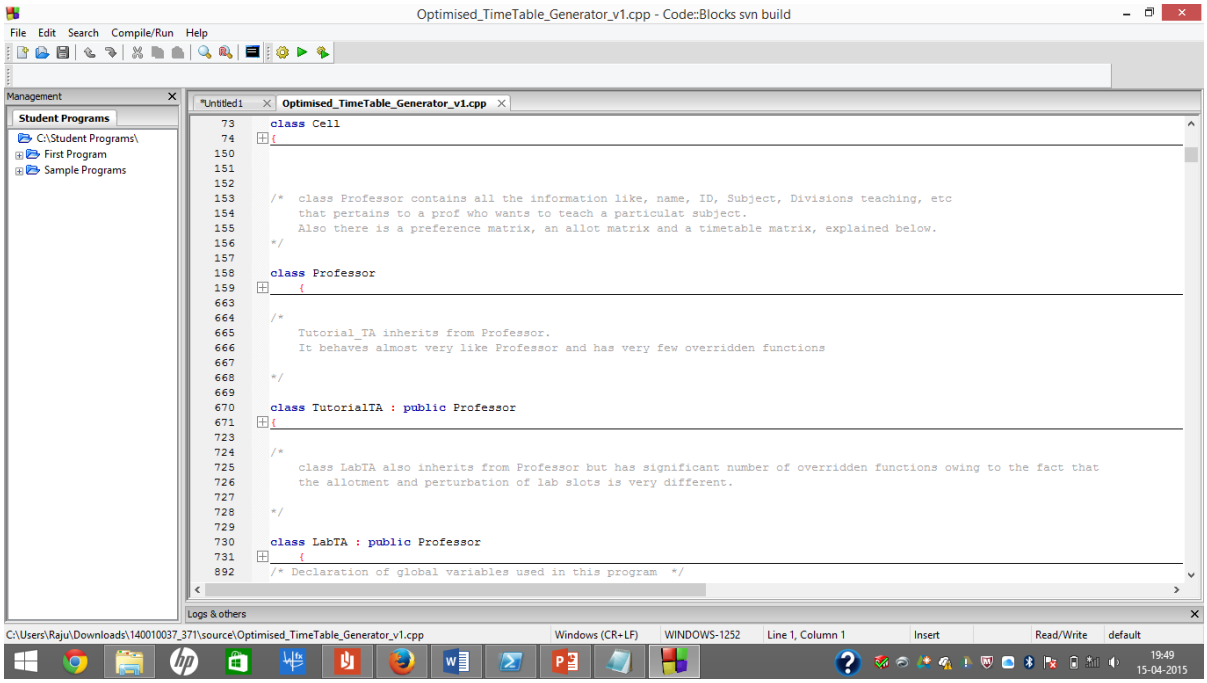
List of test-cases:

a) All professors/TAs want the same slots. In this case optimisation beyond a certain limit is not possible. An objective function (a measure of level of optimisation to be used throughout the program) will thus indicate the extent of optimisation thus done.
b) Half the professor's want slots that are ranked low in the other half's preference. In this scenario, the optimisation should be self-evident.
c) A default test case, where pseudo-random allocation of preferences happens, the objective function must also vary accordingly.
d) Professor's input is inconsistent, does not adhere to the input standards clearly mentioned in the input portal, then he would be promptly asked to enter again. Also if vacancies for a particular subject has been filled (it is on first-come-first-serve basis) then, he can either choose a different subject or exit.
e) As for generic case, any set of input must be optimised under the value of the objective function.

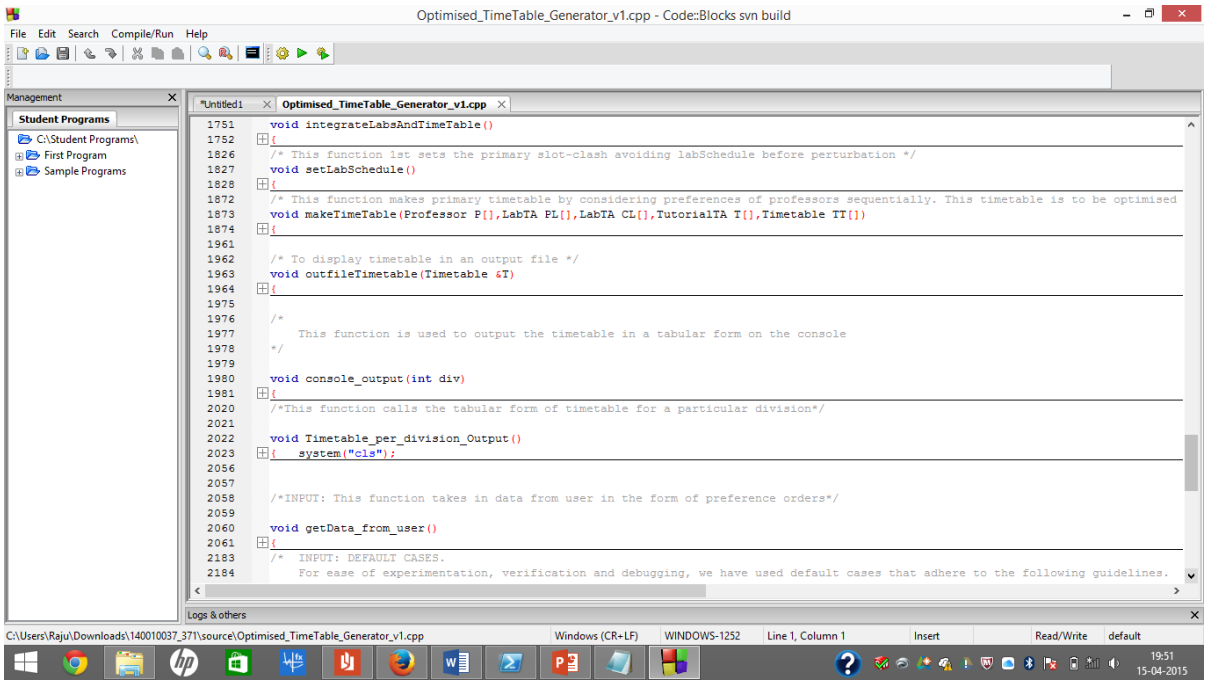
In the project development, the easy module includes generating a simple, unoptimised time-table as a base to work upon while the most difficult module to work on is perturbing the systems while keeping track of an objective function. Taking Input/Displaying Output can be listed of moderate difficulty though very crucial to link with the other modules.

5 Implementation (and features):

The entire code has been divided into a number of classes, for various entities of our program such as Professors,TAs, etc :



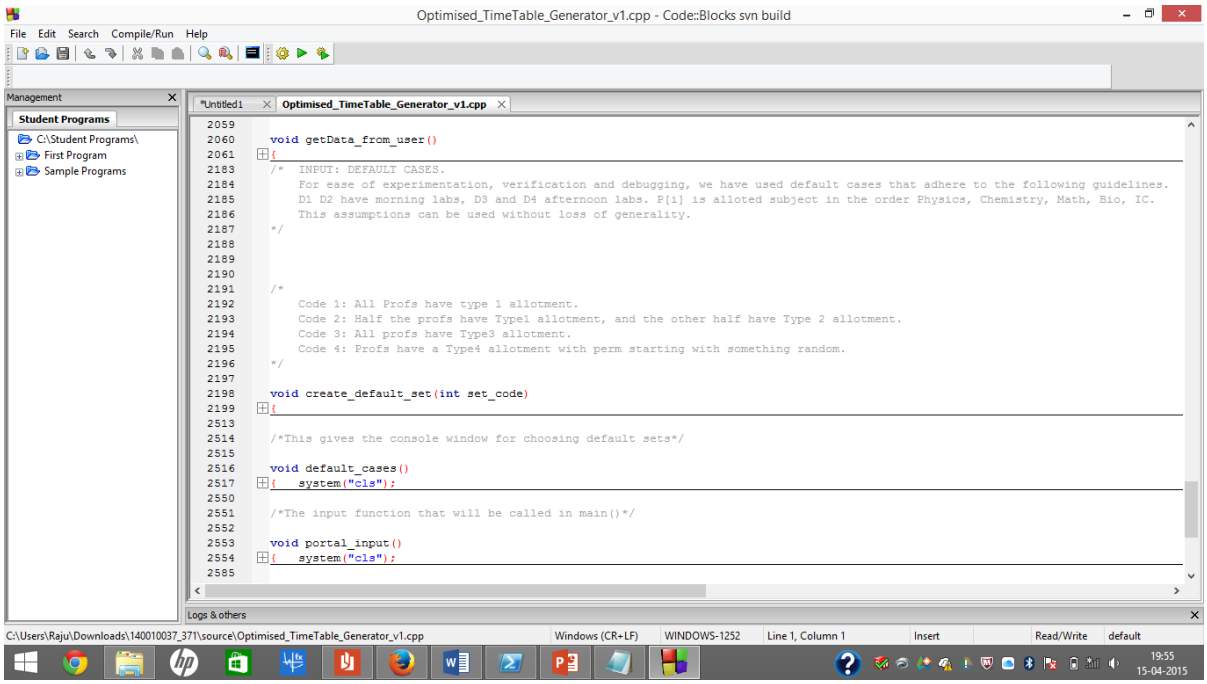
And a couple of global functions such as:



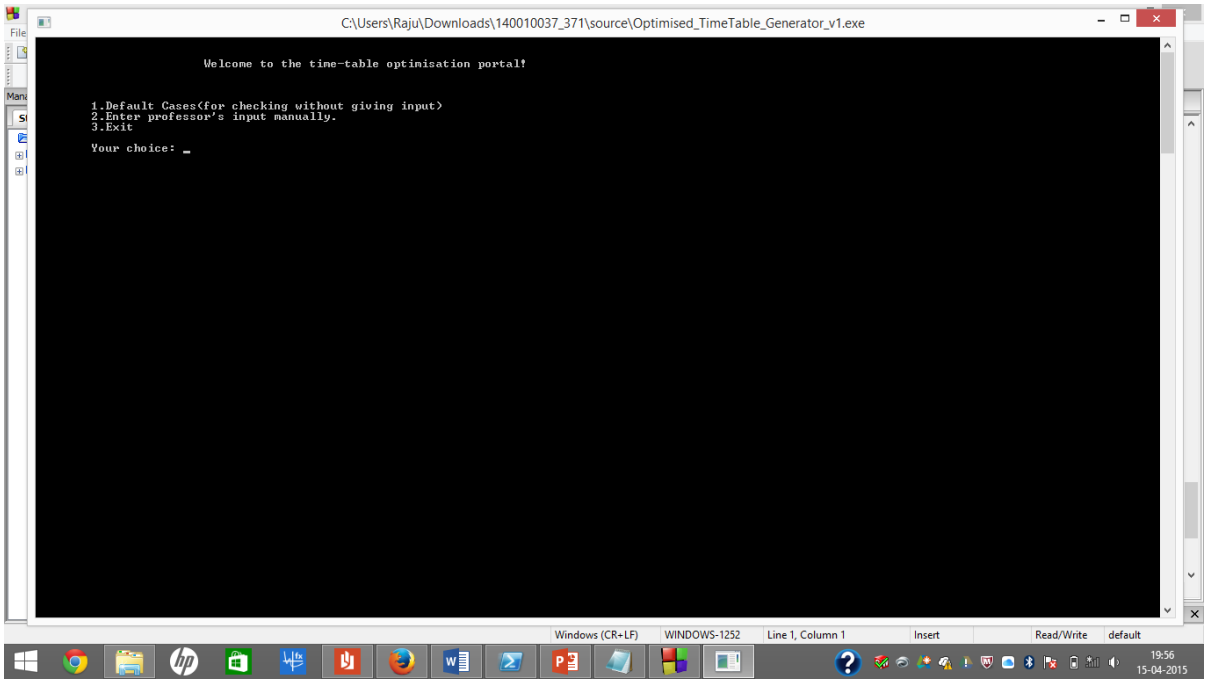
As we can see, some of classes which are common to other classes have inherited the superior class.

The input basically occurs through console where the user has the opportunity to either choose to enter all the preferences of the teachers manually or gets to choose the option which assigns the preferences of the teachers with the most likely set of values (default values which also serve as test cases).

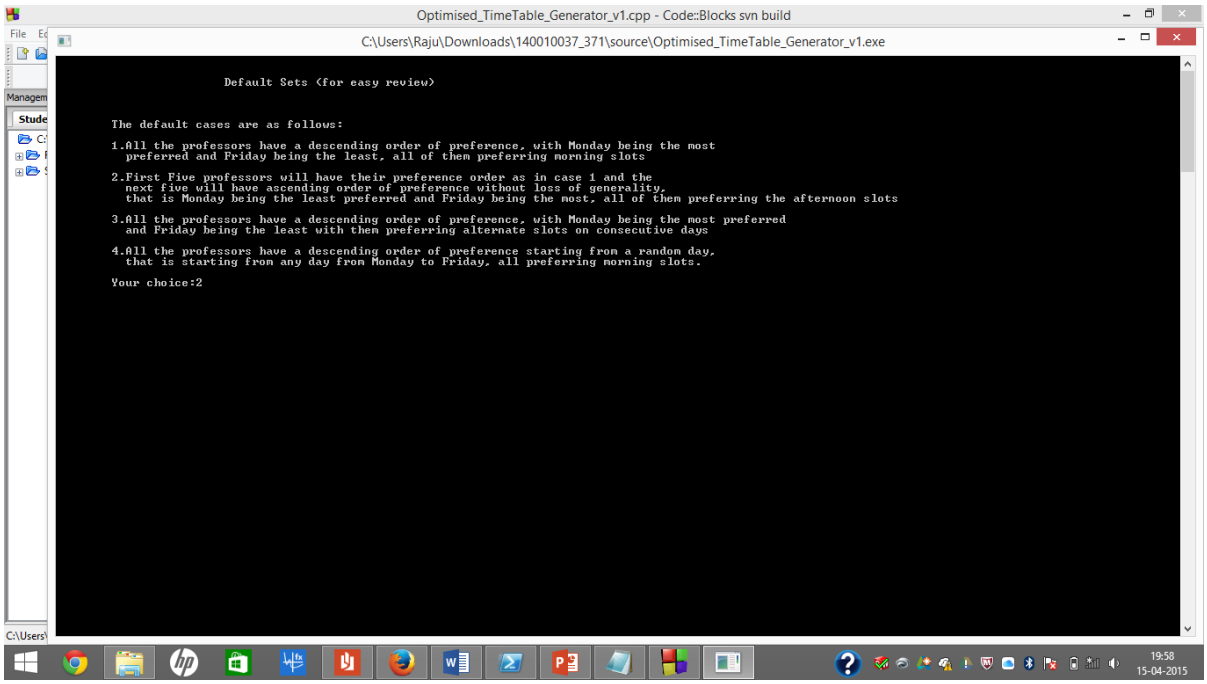
The functions concerning input are shown below:



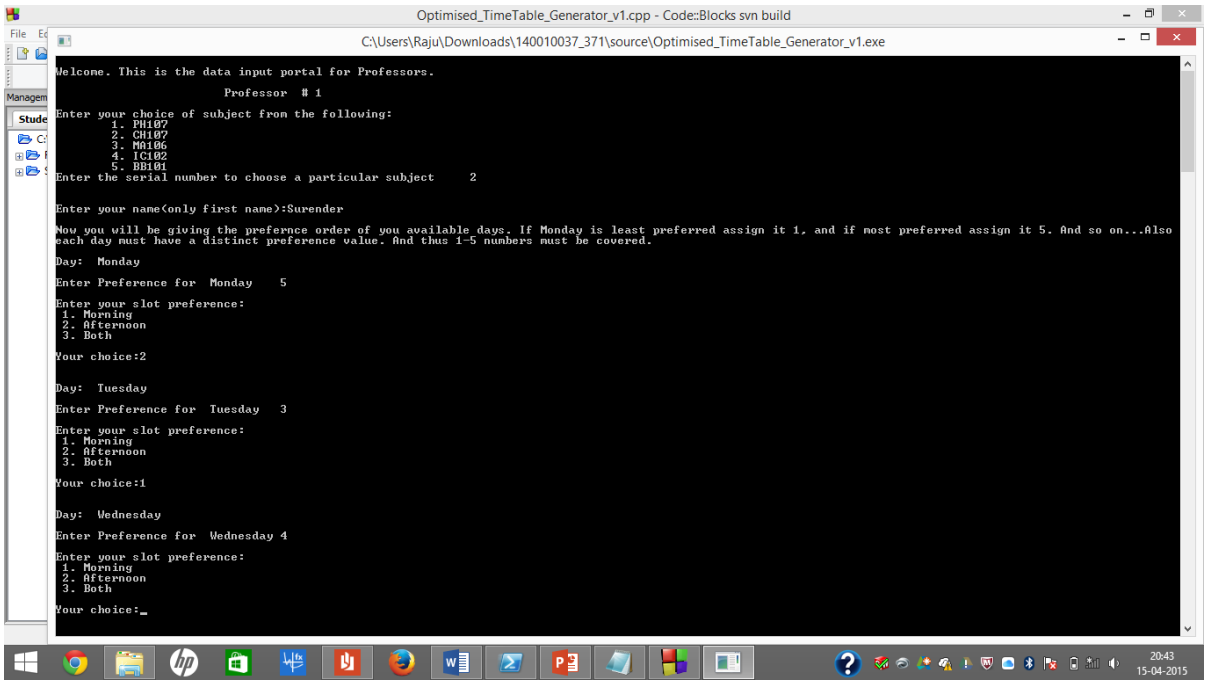
And the input on the console is of this manner:



On entering option 1, we get to choose the the default case we want.



Perhaps, we choose to go with option 2. The interface is of the following type:

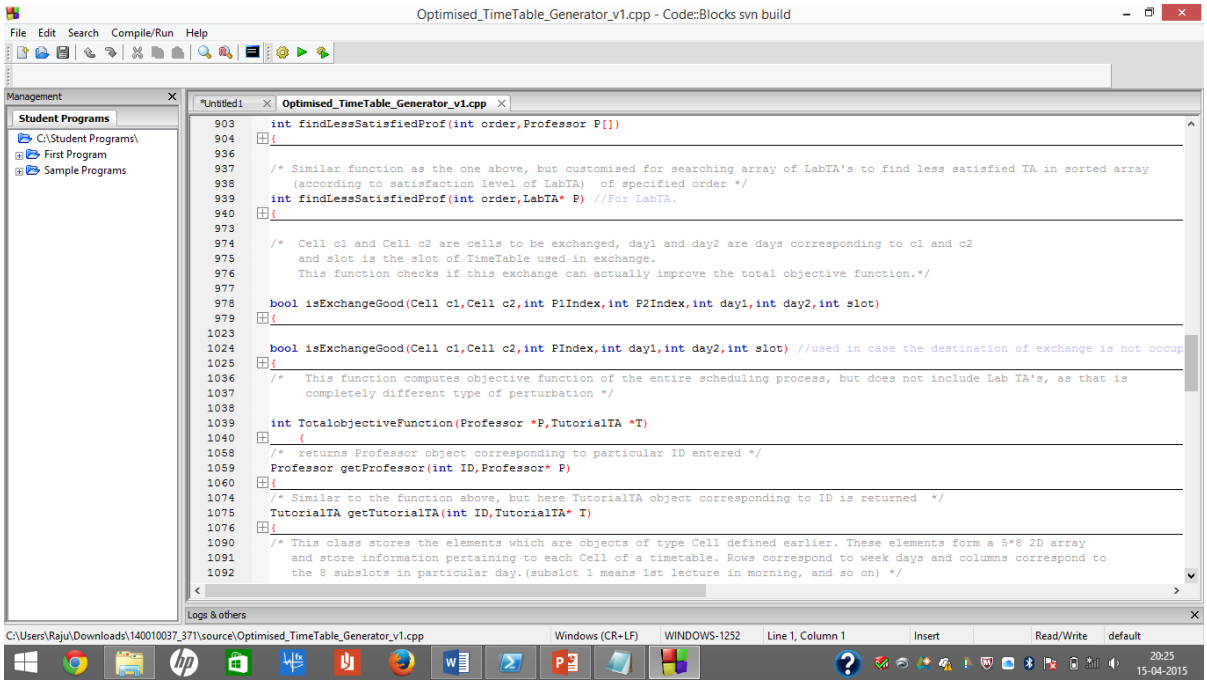


The same will be followed for all the professors and teaching assistants.

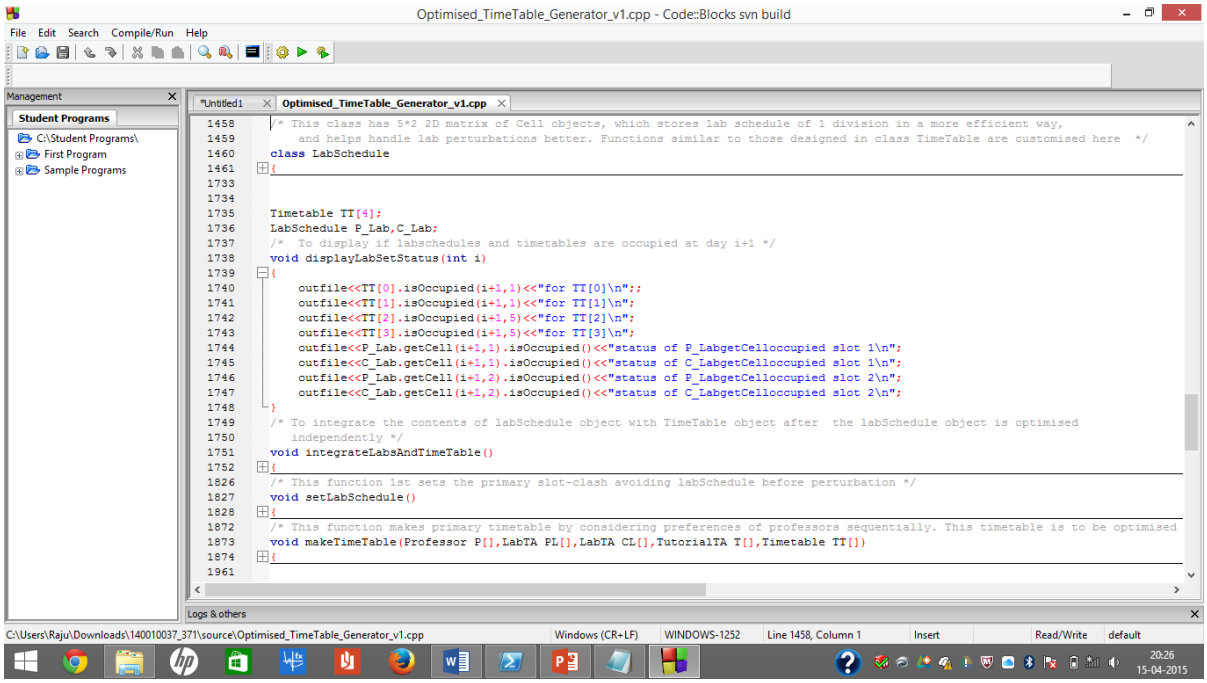
The above basically deals with input module of our code.

The algorithm module of the code is implemented in this manner:

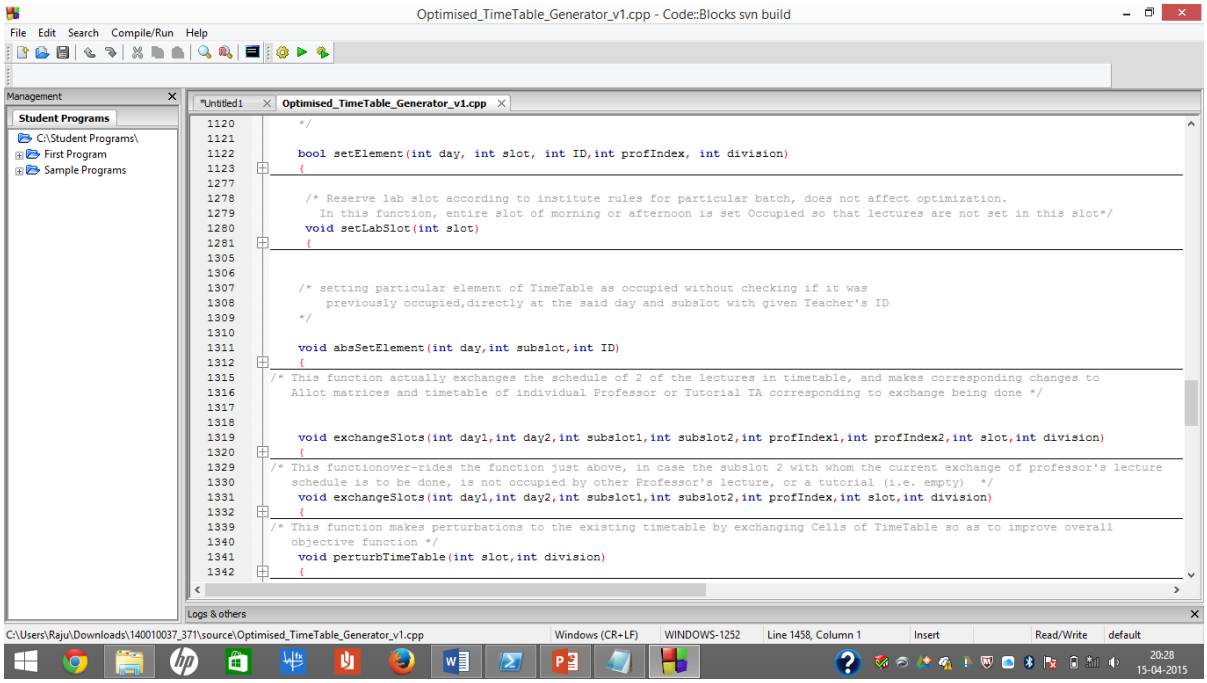
Basically our algorithm is on perturbations which has been implemented through various global functions such as:



And…



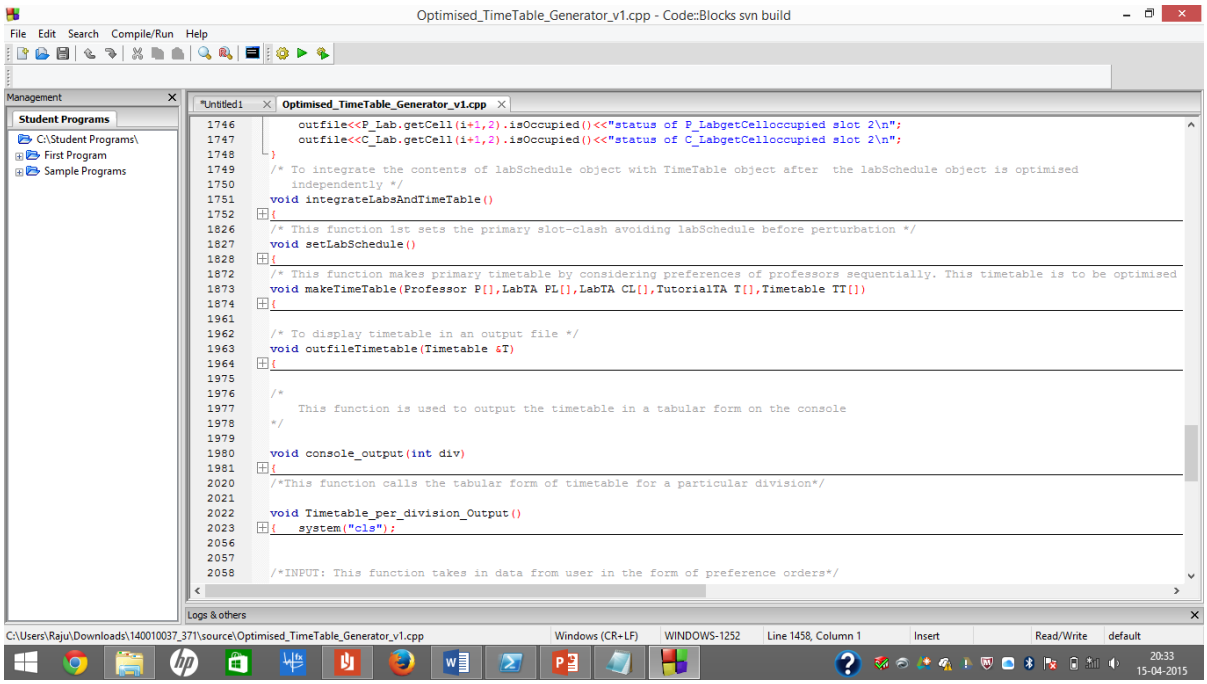
But the heart of the program, the perturbation related functions are located in class Timetable and is shown below:



The above deals with the implementation of the algorithm module.

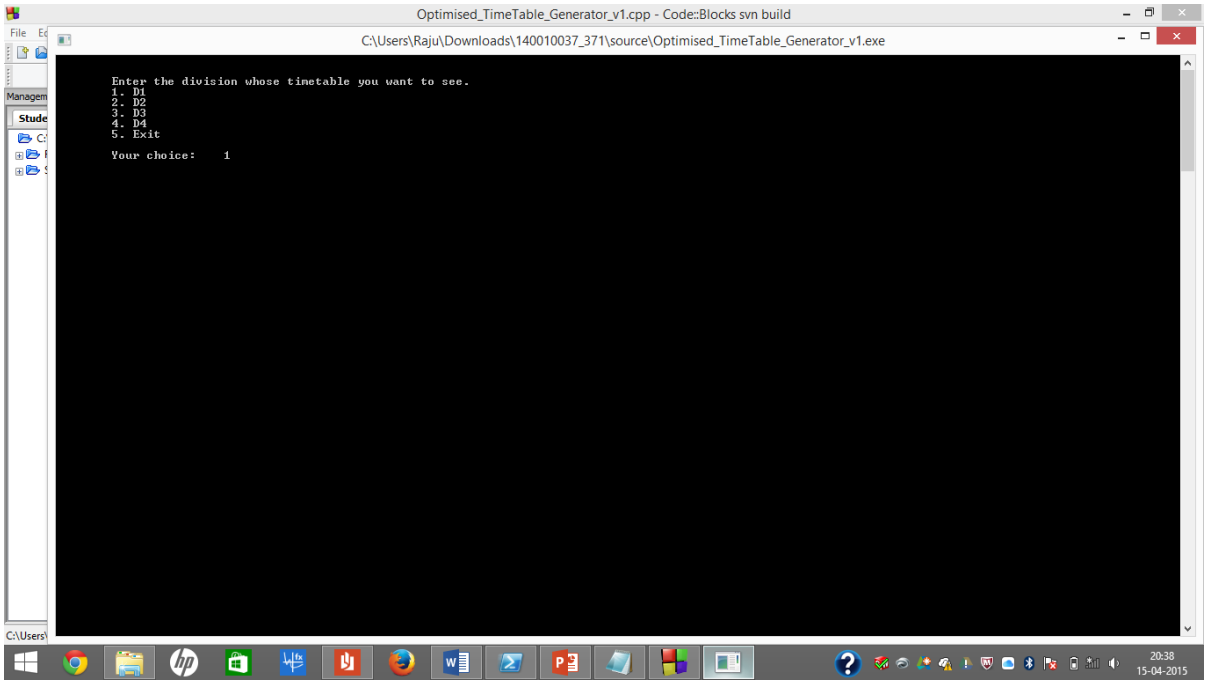
As for the output module implementation, we once again have a couple of functions as shown:

The last three of functions displayed in the below screen shot basically deals with the output display on the console:

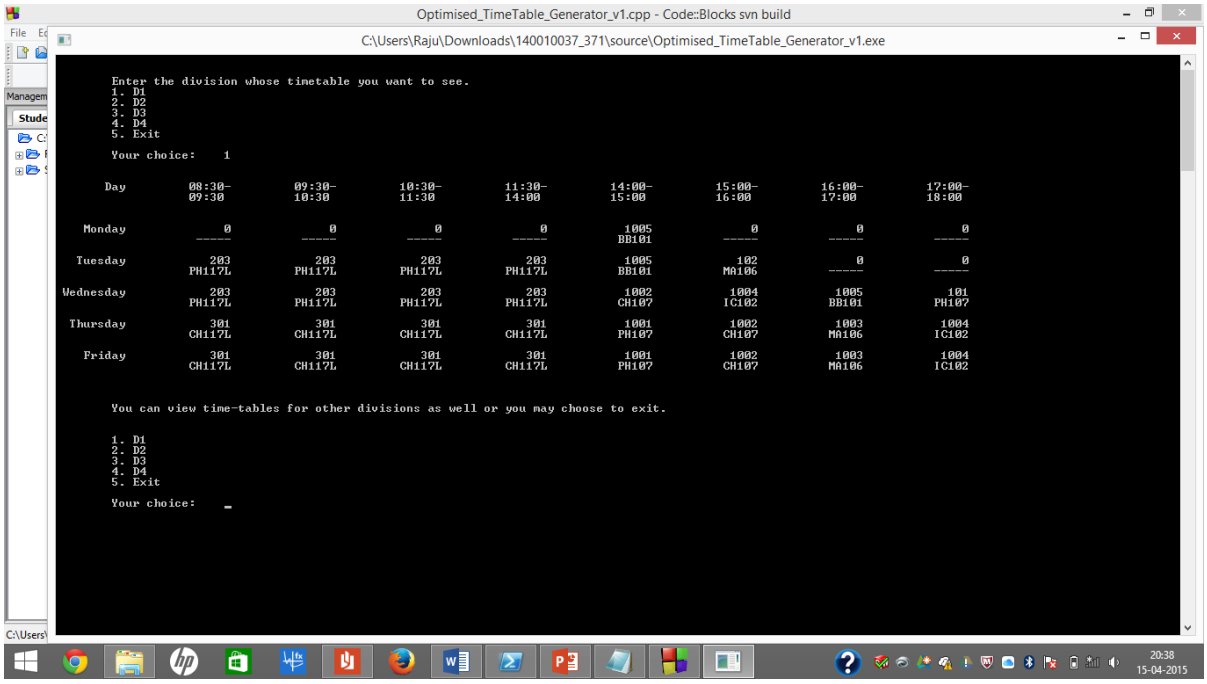


Coming back to the input we had entered initially, for the default case 2, we are asked to enter the number of the division corresponding to the time table we would wish to have.

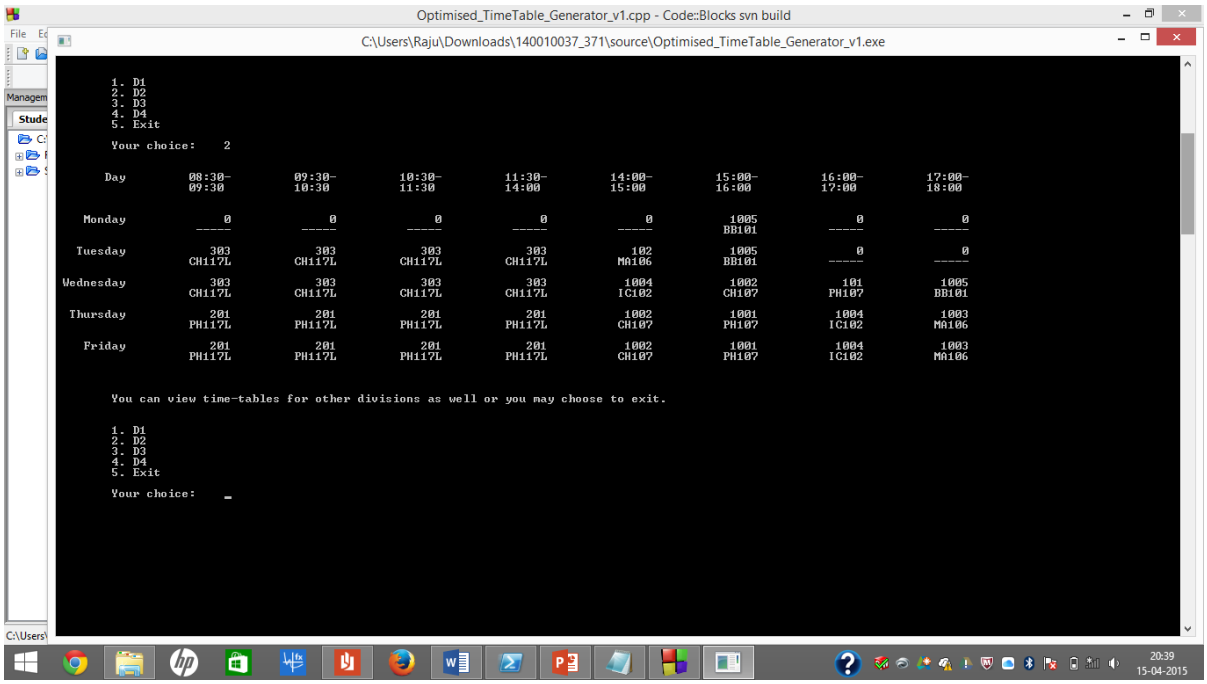
If we choose to have the timetable for division 1, we enter 1 and we get its timetable. The same follows for the other divisions.



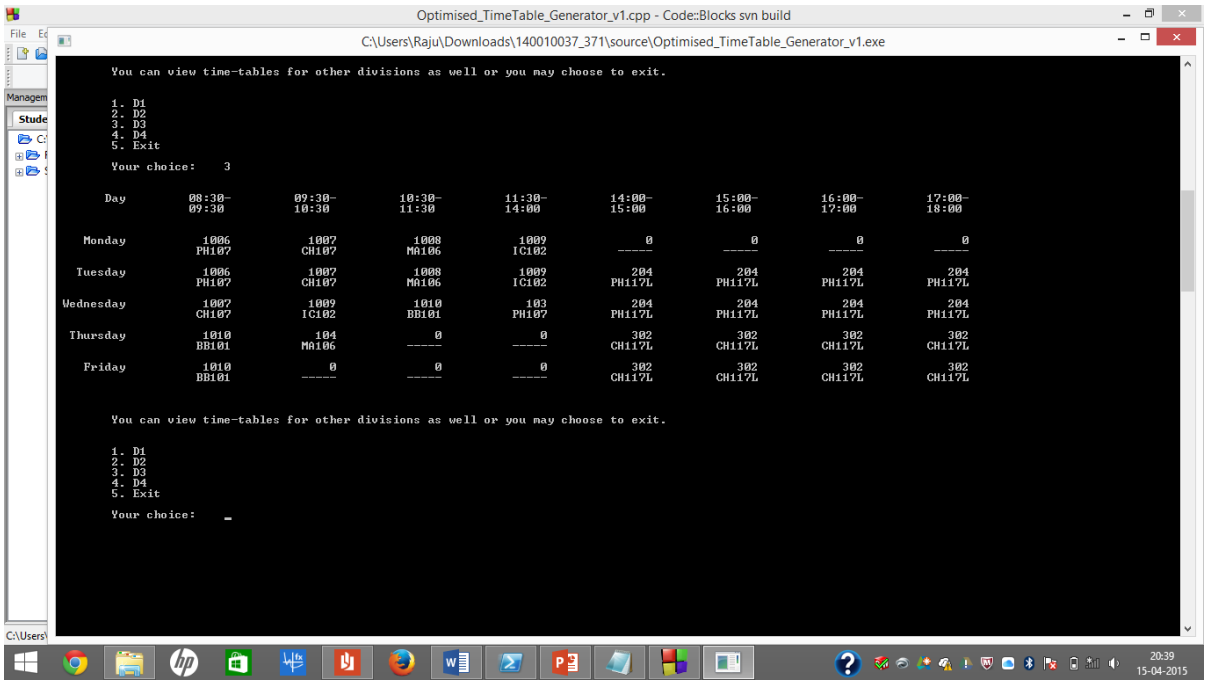
And the timetable generated is:



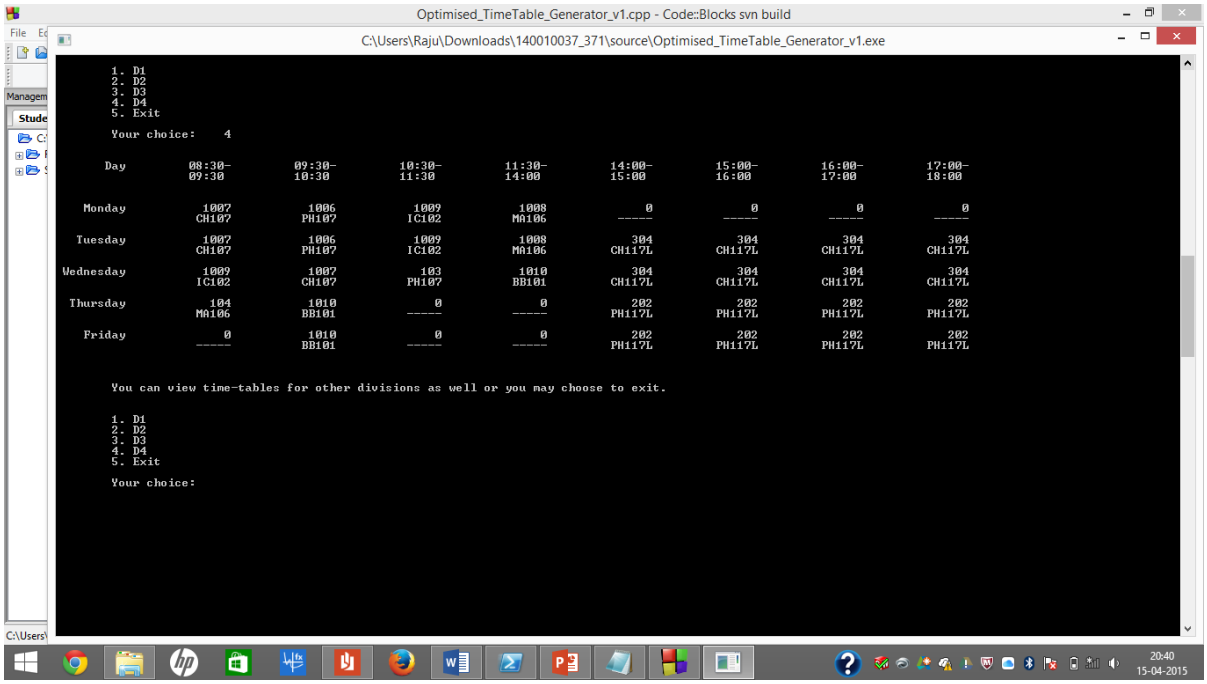
And for division 2:



And for division 3:



And for division 4:



And we enter 5 to exit.

This basically gives a broad outlook into our implementation of the program.

6 Discussing challenges faced (and how the problems were solved):

Getting the algorithm right was a tough one. Since so many functions were interlinked, coming up with a clear cut algorithm was time consuming and constant checks were needed. After a lot of trials and altering the logic of modules of code we were able to overcome the diffculties of the algorithm and received a functional result.

The most common challenge faced by any programmer is debugging. And debugging 2500 lines of code was on hell of a task. A lot of errors were occuring due to array limits going out of range and finding out each and every one of them was not easy. Syntax errors were considerably better to correct. One of the most common errors were global functions being declared as “out of scope”. This was because their definition and call positions were not appropriate. This was solved by including the function declarations at the beginning of our program.

Another set of errors occurred when we integrated all our pieces of code. Due to high interdependency among the functions, any alteration had a very high chance of resulting in an error. However, we overcame this by verifying every single line of code from various versions and correcting it into the valid one.

Even determining the test cases was a challenge. And redirecting the input into the system assigning preferences once again required a lot of labour. But after spending a lot of time and discussing it out, we settled on the default cases as explained in the implementation.

Time was also a very crucial factor and we had to sacrifice a lot to accomplish this.

7 Instructions To Run Our Code:

Make sure codeblocks has been pre-installed.

Log into moodle or github and download the compressed folder of our code.
The link to download from moodle is: http://moodle.iitb.ac.in/mod/assign/view.php?id=6533

Decompress the folder and you will be able to view our .cpp file named:
Optimised_TimeTable_Generator_v1

Open codeblocks, click on file option in the menu bar, click open and browse to open our .cpp file.

Click on complie option.

Click on run and you should definitely be able to carry out the timetable generation.

8 Future Prospects:

We can attempt to create a timetable for the entire institute which would include a lot of courses and variety of different slot clashes, so obviously it would include a variety of variables. The number of students registered for each course will vary. Also, one professor might be able to teach different courses (more than one) and he can teach for students belonging to different years of engineering.

The slots clashes will once again have to be managed in this case, in a more diligent manner ofcourse.

We can also include constraints such as distance between lecture halls of consecutive classes for students to commute in a comfortable manner.

Also, we can generate additional department timetables too, along with the teachers' and student' timetables too

The algorithm too, will have to be improved to manage the above scenario in a more efficient manner.

9 Conclusions

This project was done under the time-constraint of only effectively having half a semester. The future prospects for this project can be easily extended from the existing program and the efficiency of the code can also be further increased. We hope to implement such improvements later.

10 References

Code::Blocks Installation Guide
http://wiki.codeblocks.org/index.php?title=Installing_Code::Blocks

GitHub Repository:
https://github.com/BasuhiRavi/140010037_371