Objective

To program a timetable generator that would generate timetables for different sections and teachers. The program would take the details of the sections and the teacher, including their optional subject and create timetables that ensure multiple constraints are taken into account such as teacher availability, lab availability, ensuring teachers don’t have more than 6 classes per day etc.

Problem Definition

Wherever any educational system with multiple classes and teachers exists, there must always be timetables. Timetable dictate the way in which the day flows for students and teachers alike and thus are critical to the functioning of the school. As such, to ensure the school functions smoothly, timetables not only have to prevent clashing schedules but must also ensure that the timetables of both students and teachers must be spread out across the week. This helps avoid excessive exertion and increases productivity of the school.

Up until 20-30 years ago, however, all the timetables had to be made by hand due to the lack of any technological capabilities that could take into account all the different constraints. However, with the advancement of technology it is now possible to create programs that can effectively create timetables. The various drawbacks of generating timetables by hand are:

1. Timetable generation becomes a mammoth task that can take days to complete, resulting in lost productivity and wasted time as opposed to computers which take seconds to do the same job.
2. Timetables could not address all the concerns of students and teachers which could result in too many periods for a teacher in one day.
3. Human error can result in certain classes not having wrong subjects which can cause confusion

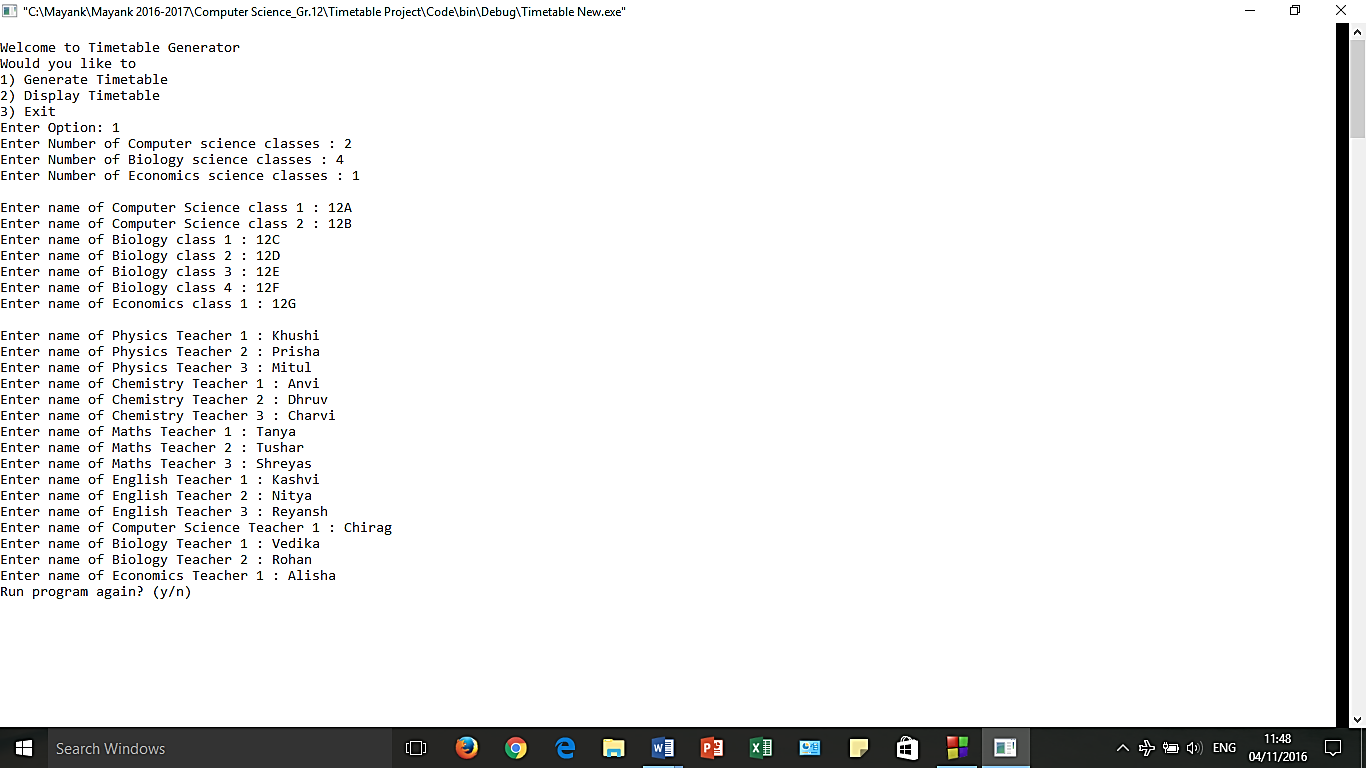
Through our program, we aimed at generating timetables for the science sections by cyclic assignment which adhered to the following constraints:

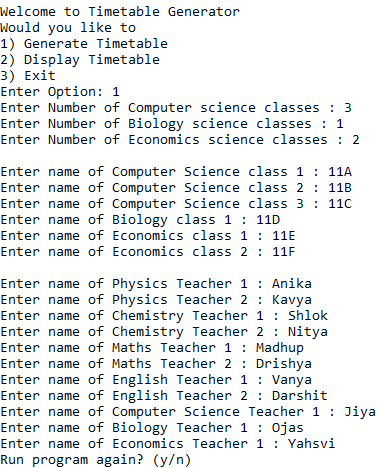
* Each section must have the following periods in a week

|  |  |  |  |
| --- | --- | --- | --- |
| Subject | No of periods/week | Minimum number of periods in a day | Maximum number of periods in a day |
| Physics | 7(Theory)+2(Lab) | 1 | 2 |
| Chemistry | 7(Theory)+2(Lab) | 1 | 2 |
| Math | 9 | 1 | 2 |
| English | 7 | 0 | 2 |
| Comp/Bio/Eco | 7(Theory)+2(Lab) | 1 | 2 |
| PT | 2 | 0 | 2 |

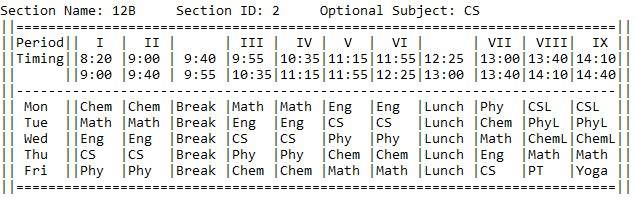
* Each teacher taught at max 3 sections
* Each teacher had no more than 6 classes/day
* Each teacher taught 7 (English) or 9 classes (All other subjects) per section in a week
* Each section has a block of PT/Yoga per week
* Wherever possible, the classes are grouped in blocks of 2 periods
* Block periods must not occur across lunch or break
* There is only 1 lab per subject and that must be free for use by different sections

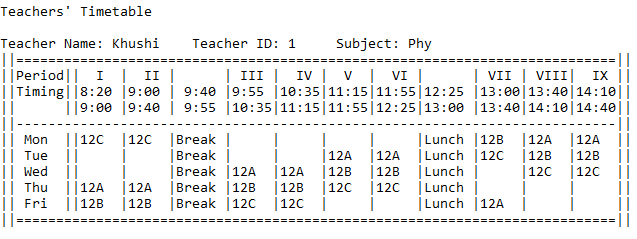
Analysis:

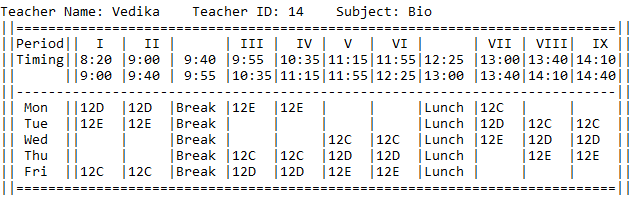
Sample Inputs:

1) 2)

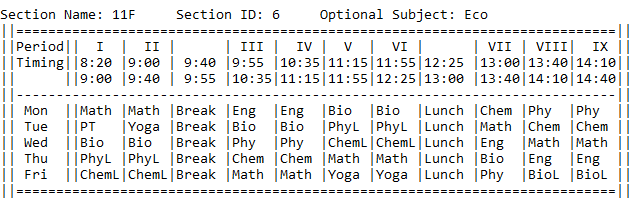
Respective Outputs:

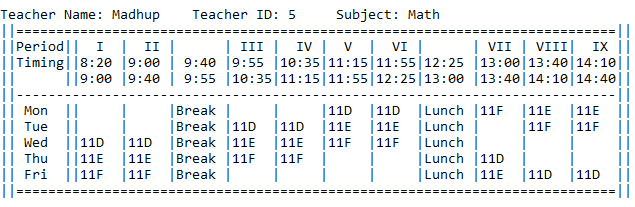
1)





2)



Design

First each section is assigned teachers for subjects taught linearly.

Then this application populates the timetable in a cyclic fashion. For example

Section A timetable (assume it is a computer science section)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | Physics | Physics |
|  |  |  |  | Physics | Physics |  |  |  |
|  |  | Physics | Physics |  |  |  |  |  |
| Physics | Physics |  |  |  |  |  |  |  |
|  |  |  |  |  |  | Physics |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Chem | Physics | Physics |
|  |  |  |  | Physics | Physics |  | Chem | Chem |
|  |  | Physics | Physics | Chem | Chem |  |  |  |
| Physics | Physics | Chem | Chem |  |  |  |  |  |
| Chem | Chem |  |  |  |  | Physics |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Math | Math |  |  |  |  | Chem | Physics | Physics |
|  |  |  |  | Physics | Physics | Math | Chem | Chem |
|  |  | Physics | Physics | Chem | Chem |  | Math | Math |
| Physics | Physics | Chem | Chem | Math | Math |  |  |  |
| Chem | Chem | Math | Math |  |  | Physics |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Math | Math | Eng | Eng |  |  | Chem | Physics | Physics |
| Eng | Eng |  |  | Physics | Physics | Math | Chem | Chem |
|  |  | Physics | Physics | Chem | Chem | Eng | Math | Math |
| Physics | Physics | Chem | Chem | Math | Math |  | Eng | Eng |
| Chem | Chem | Math | Math | Eng | Eng | Physics |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Math | Math | Eng | Eng | CS | CS | Chem | Physics | Physics |
| Eng | Eng | CS | CS | Physics | Physics | Math | Chem | Chem |
| CS | CS | Physics | Physics | Chem | Chem | Eng | Math | Math |
| Physics | Physics | Chem | Chem | Math | Math | CS | Eng | Eng |
| Chem | Chem | Math | Math | Eng | Eng | Physics | CS | CS |

Notice how the new periods assigned follow a pattern of cyclic arrangement.

Also now Eng needs 9 periods a week and Lab needs to be assigned so convert last possible period in a day when lab is free to Lab period or in case of English which has 7 periods a week make such an available slot = PT/Yoga

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Math | Math | Eng | Eng | CS | CS | Chem | Phy(L) | Phy(L) |
| Eng | Eng | CS | CS | Physics | Physics | Math | Chem(L) | Chem(L) |
| CS | CS | Physics | Physics | Chem | Chem | Eng | Math | Math |
| Physics | Physics | Chem | Chem | Math | Math | CS | PT | Yoga |
| Chem | Chem | Math | Math | Eng | Eng | Physics | CS(L) | CS(L) |

The program has 3 classes: “Section,” “Teacher,” and “Lab,” described below:

1. Class “Section”:

Data Members:

char name[5]; --> Name of the section

int id; -->Section Id : Assigned by program for class assignment

int tt[5][9]; --> Section Timetable: Subject code stored in a 5 by 9 array

int sub[10][2]; -->Subject: Stores the subject code and teacher id

Member Functions:

Section(); -->Class Constructor

void getSub(int code); -->Assigns subject codes of the section

int hasSub(int a); -->Function to check if particular subject is present

void getId(int x); -->Assigns id as it is a private member

int putId() -->Functions returns the section id as private member

1. Class “Teacher”:

Data Members:

char name[20]; --> Name of the teacher

int id; -->Teacher Id : Assigned by program for class assignment

int sub: -->Subject the teacher is teaching

int tt[5][9]; -->Timetable of Teacher: Section id stored in a 5 by 9 array

int noOfSectTaught;-->Stores number of section taught by teacher

Member Functions:

Teacher(); -->Teacher Constructor

int isFree(int d, int p); -->Function to check if teacher is free

1. Class “Lab”:

Data Members:

int sub: -->Subject of the lab

int tt[5][9]; -->Timetable of Lab: Section id stored in a 5 by 9 array

Member Functions:

Lab(); -->Lab Constructor

void getSub(int x] -->Assigns subject as it is private member

int putSub() -->Returns the subject as it is a private member