OOM Lab Assignment 1

1. Make a **student** class with attributes roll number and name (single word only). Read input from the console and print details on the console. The input first consists of the number of test cases, then each test case has two strings, roll number and name.	student studentCollection
	TA TACollection
2. A class consists of an **array of students**. Read all student details in a class. *Sort* the class roll number wise. *Print* the class sorted as per roll number. The first input is the number of test cases, with each test case followed by the number of students and details of all students. You should write your own sorting algorithm and not use the collection framework sorting utilities.	studentWithTA studentWithTACollection assignment attemptedAssignment assignmentStudent assignmentStudentCollection
3. The institute provides **TAs** for all students to 'take care' of the **students**. Every student gets a TA. As per a new policy of the institute there is 1 TA for every *k* students. Assign TAs to students roll number wise. The first TA as per input gets the first *k* students, the 2nd TA gets the next *k* students and so on. The last TA gets all the left over students that may be less than, equal to or greater than *k*. Please make sure that every TA and student should only be stored once in the memory, and not be copied along with every student.	instructor
4. In the above question also print the students associated with every TA separately. Please make sure that every TA and student should only be stored only once in the memory, and not be copied along with every student.	
5. A student attempts numerous **assignments** and gets **marks** for the different assignments. In question 2, the student details are followed by the assignment details. Print the total marks with breakup obtained by each student in addition. Note a single space between every entity.	

OOM Lab Assignment 3

1. An institute has examinations going on and therefore **question papers** need to be made. The institute has a poor history of **students** copying from each other and therefore every student is given a separate question paper made from a **question bank**. Every student should get exactly *k* **questions**. Every question can be given to a maximum of 1 **student**. So, no two students can have any question common among themselves. The question papers are made by a **TA**, on behalf on an **instructor**. The affinity of a question to a student is defined by the absolute difference between the question number ID (integer) and the numeric form of the roll number (of the form 1234567, where abc are characters and 1234567 can be any digit). In case of a tie a smaller question number (primary) and a smaller roll number (secondary) are preferred. Print the question paper for every student in the order that the students enter the room.	question student TA Greedy StudentWise QuestionWise affinityFun closenessLoving bestRollNumber lucky questionStudentAssignment instructor
2. In question 1 consider that there are *n* **TAs** instead of just one. All TAs work under the instructor. Every TA has a different affinity function. There are typically 3 types of affinity functions: • ***closeness loving*** (CL) TAs, who use the absolute of differences just like question 1; • ***best roll number*** (BR), who attempt to go roll number-wise with the least roll number first • ***lucky* types** (LU), who have a lucky number *l* and define affinity as (numeric part of roll number + question ID) mod *l*	
The instructor asks every TA to give a question-student pair and makes the **assignment**. The first assignment is based as per the 1st TA, the 2nd assignment as per the 2nd TA and so on in a round robin manner. Give the question papers hence formed.	
3. Assume in question 2, the instructor assigns students to TAs and then each TA allocates questions as per its own methodology. The instructor assigns the students to TAs in a round robin format. There are three types of TAs. • **Greedy** (G): These TAs work by finding best pairing question and student and hence do the allotment. The metric used is the affinity function and the tie breaking condition is question number (primary) and roll number (secondary) • **Student-wise** (SW): These TAs go roll number wise for students and find the best question for the student. The second student is assigned only if the first cannot be	

assigned and so on. The students should be taken in a sorted order. • **Question-wise** (QW): These TAs go question wise and find the best student. The second question is assigned only if the first cannot be assigned and so on. The questions should be taken in a sorted order. Since no question can be repeated, the allocation is again done in a round robin fashion.	

OOM Lab Test

1. It is (nearly) time for Effervesence and the **events** are being geared up. The events consist of event ID, name, description, day (1, 2, 3, etc.), time of start and duration. You are given the details of ***n* events**, print “Day 1” followed by all day 1 events in the order of their start, “Day 2” followed by all day 2 events in the order of their start, and so on. In case two events on the same day start at the same time, the printing should be in alphabetical order of event ID. You should not assume that the event is only of 3 days. 2. A **student** registers for numerous such events. The registration information for every student is provided by providing the roll number and event ID. Given a number of such **registrations**, first print the **event schedule** as in question 1, however printing all participating students in a new line. Then a number of queries are raised with every student specifying the roll number of a student. Print the schedule of the queried students. 3. In question 2 add the constraint that the student cannot register in two overlapping events and attempting to register for an event that causes a conflict results in failure in registration and the registration is not done. No event in the input shall start before midnight and end after midnight. 4. In question 3, assume that there are 3 types of events: The **general events** are open for all, the **course specific events** are specific to some course types, the **level specific events** are specific to some levels of study only (ug, ph or phd). Check eligibility before carrying the registrations. Use inheritance.	student event course level festival

OOM Assignment 2

OOM Assignment 2

Part 1

A **time table** for this institute has the following **slots** for taking classes: 9-10, 10-11, 11:15-12:15, 12:15-1:15, 3-4, t4-5, and 5-6. The time table spans from Monday to Friday. Make the time table for 1 batch.

Each **course** has a number of **slots** with durations. Like IOOM332C has 2 slots in the time table, one for 2 hours and one for 1 hour. The slots must be strictly such that the entire duration of the slot must be available as continuous. So a 2 hour class cannot be taken as 10:00-11:00 and 11:15-12:15. The theory, labs and tutorials are entered as separate courses with different course codes.

The allocation of time table for every course happens in a prioritized manner. All courses have a priority which is an integer. The lower is the numeric priority value, the more is the importance of the course. You first allocate the first slot associated with the highest priority course. Then you allocate the first slot associated with the second highest priority course and so on till all courses have one slot filled up. Then you allocate the 2nd slot associated with every course in the same order of priority, and so on.

Every slot has a preference, which is both in terms of the time of the day that a course should preferably run (primary) and the day on which the course should run (secondary). The day preference is specified by a string 143 means that the preference is 1 (Monday), followed by 4 (Thursday), followed by 3(Wednesday), and thereafter any day which are all equally good. The time preference is specified by a string where every 4 consecutive integers denote a time so a preference 090010001115 means, in increasing order of preference, the slots, 09:00-10:00, 10:00-11:00, 11:15-12:15, and so on.

While allocating a course, you first try to find the best slot available as per the primary and secondary time and day preferences. If none of the preferences are available, then the allocation takes place so as to balance the workload, and the day of the week with the least number of working hours is selected (primary), in the preference order from Monday to Friday (secondary). The selection of time is done so as to have classes as early as possible, and the first feasible and available slot is selected.

faculty
course
slot
timetable
day
TimeTablePopulator

Utility: time

Consider requested time slots are (for a 1 hour class) 10-11 and 11:15-12:15, while the requested day slots are T and W. Consider the total working load so far is M: 1, T: 3, W: 4, Th: 6, F: 2. So a typical day preference as per the time table is M(1), F(2), T(3), W(4), Th(6). The following are the attempts made to place a course:

If a course cannot be allocated due to no available slots, then, it is notionally added on a Saturday. The Saturday courses have no time specified and are taken as per the wish of the instructor. All Saturday courses should be appended in a sorted order, sorted by course code.

Part 2

There are more than one **batches** running in this institute. You need to make a time table for all of them, instead of just one. Ensure adherence of the constraint that no faculty can simultaneously take two classes, and no student can simultaneously attend two lectures.

Part 3

In part 2, print the **faculty** time table for all faculty in the same order as supplied in the input (which may not be the alphabetical order).

Part 4

The time table was floated to all faculty and the faculty gave some **suggestions** that need to be incorporated into the time table, if admissible. Each suggestion has a priority and needs to be incorporated in the same order of priority. The suggestions include the course code, slot number and new preference in day and time format. The allocation is done as per the same principles as above and may result in a slot being placed in a poorer location as before when none of the preferences are met.

Order of suggestion inputs is code, slot number, priority, day preference and time preference