Specification for each required feature. The specification should include a description of input parameters and output (usually screen outputs), and an example of how a user can use this feature (e.g., exec XXX(…) where XXX is the procedure name). You don’t need to implement any of these features at this point.

**Feature 1: Add a new course.**

Input: course name, program id, number of credits, grading format, whether the course is required, room type, number of sections.

Output: If a course with the same name exists in the program, update the existing course with the values of the input parameters. Otherwise, insert a new course, generate a new course id and print out the new course id.

Exec add\_course();

**Feature 2: Add a new instructor.**

Input: instructor name, department id, and instructor type (full time or part time).

Output: If an instructor with the same name exists at the same department with same instructor type, print an error message. Otherwise insert the instructor and print out the generated instructor id.

Exec add\_instructors();

**Feature 3: Allow an instructor to enter teaching preferences for a given year and semester.**

Input: Instructor id, year, semester, course load (the number of courses the instructor needs to teach and multiple sections of the same course count as one course), a list of courses the instructor is willing to teach (this list should be greater or equal than the number of courses need to teach), for each such course, the number of sections the instructor is willing to teach, and up to two days of week that the instructor cannot teach.

Output: Check special cases including 1) the input instructor id is valid. 2) the list of courses is smaller than the course load, 3) the number of blackout days is over two. Print an error message if one of these cases occurs. Otherwise insert these input to appropriate tables.

Exec teaching\_pref()

If instruct\_id = instructors.instruc\_id then

If list of courses = course\_load then

If cannot\_teach\_days >2 then

Add preferences;

Else

Dbms\_output.put\_line(‘error’);

**Feature 4: Allow a student to search for all courses offered in a given year, semester, and program.**

Input includes year, semester, program id.

Output: for each scheduled course section the name of the course, number of credits, grading format, schedule id, section id, name of instructor, name of classroom, days of class, start and end time of each class, and whether the class is open or full. Please order the results by course id and section number. Please also check whether input program id is valid.

Exec if program\_id = program.program\_id then

Dbms\_output.put\_line (course\_name, credits, grade\_format, schedule\_id, section\_id, instruct\_name, room\_name, time\_day, time\_start\_time, time\_start\_time+time\_length (end\_time), section.status);

Order by course\_id, section\_id;

Else

Dbms\_output.put\_line(‘error’);

**Feature 5: Compute the number of assigned courses for all instructors given a year and semester.**

This feature checks the number of scheduled courses taught by each instructor for that year and semester and store this number in the course load table. Note that multiple sections in the same course is counted as one course. This feature will be called by other features.

**Feature 6: Assign the course to instructors who are willing to teach that course.**

Input Includes Course ID , Year, Semester

Output: First check whether the course needs more sections (compare number of sections needed for the course from the course table to the number of sections already scheduled). If so assign new sections to instructor who is willing to teach that course and create schedule for this course (you can leave classroom unassigned). Otherwise print a message there are enough sections.

You need to make sure that   
1) enough course sections are assigned (the number of sections is in course table). Print a message if this is not possible.

2) only instructors who are willing to teach that course will teach the course.

3) The instructor's course load is not exceeded.

4) the new schedule's capacity equals section size of the course and waiting list size is 10.

Exec Set I();

If Set I is Null then

Dbms\_output.put\_line(‘Not Enough sections are assigned due to lack of faulty’);

else

Desc Set I;

If I < Course\_load then

Exec add\_course();

Exec course\_schedule();

If there are less instructors for sections then

dbms\_output.put\_line(‘There is not enough instructors’);

**Feature 7: Assign courses in a department for a given year and semester.**

Input: call feature 6 to assign a course to instructors.

Output**:**  check whether all instructor has been assigned enough courses and print out the names of instructors who have not.

Exec Procedure Course\_Schedule();

If Instructor < Course\_load then

Add electives

If instructor < Course\_load then

Dbms\_output.put\_line (‘Instructors who have not assigned enough courses’ || Instructor)  
 else

Dbms\_output.put\_line (‘All instructor has been assigned enough courses’);

**Feature 8:** Assign room and time to a scheduled section

Input: Schedule ID

Output: First check whether the schedule id is valid. If not print an error message. Next check whether the scheduled section already has a room and time block. If so print an error message saying that the course is already assigned. Otherwise find a room and a time block pair that satisfies the following conditions:

1) The day of week is not one of the blackout days of the instructor.

2) The instructor is not teaching at that time block.

3) The room has no class scheduled at that time block.

4) If the course has multiple sections, no other section is at the same time block.

5) If the course belongs to a graduate program, choose a time block that is after 4:00 pm.

6) The number of seats in the room is greater than or equal to class capacity

7) If the course requires in a computer lab, the room must be of computer lab type.

Exec Assign\_room(schedule\_ID\_input)

If Schedule\_id\_input=Schedulde.schedule\_id then

Exec Check\_Room and Time\_Block;

IF Check\_room and Time\_block exits Then

Dbms\_output.put\_line(‘Course is already assigned’);

Else

Exec Assign\_room and Time\_Block

**Feature 9:** Assign rooms and time blocks for courses in a department given a year and semester.

Input: department id, year, and semester.

Output: This feature can call feature 8 to assign a section to a room and time block. This feature will go through every scheduled section in that department, year, and semester and assign for each section a room and time block. Please first schedule graduate courses and then undergraduate ones. For courses are all graduate or undergraduate, schedule required courses first.

Exec Assign\_room\_timeblock(Department Id,Year, Semester) ;

Exec Procedure Assign\_room();

**Feature 10:** Enter special permission for a student and a scheduled section

Input:student id and schedule id

Output: First check whether both are valid. If not print a message. If both are valid enter special permission to special permission table. The special permission includes 1) enroll in a closed class. In this case the student can enroll even if the class is full; 2) enroll without prerequisite. In this case the student can enroll even if the student has not taken prerequisite.

Exec if student\_id and schedule\_id=Vaild then

Add Special Permission(1 or 2)

Else

Dbms\_output.put\_line(‘Student id and schedule id not valid’);

**Feature 11:** Check prerequisite of a class for a given student

Input: schedule ID and a student ID

Output: Returns one if the student has taken prerequisites of the scheduled course, 0 otherwise. Please check if the schedule ID is valid first.

Exec

If schedule\_id=valid and Student has taken prerequisites then

Dbms\_output.put\_line(‘1’);

Else

Dbms\_output.put\_line(‘0’);

**Feature 12:** Allow a student to register a course given a schedule id.

Input: Schedule Id

Output: Registration of a course

If Schedule\_id and Student\_id <> valid then

Dbms\_output.put\_line(‘Invalid ID’)

If Student\_id has taken same course before and have <> got below D then

Then

Dbms\_output.put\_line(‘Student cannot take the course’);

If Student\_id has not taken prerequisite

Then

Dbms\_output.put\_line(‘Cannot Enroll’)

If Class\_Status=1 or Special\_permission\_status=1 then

Add student(student\_ID)

Update registration\_staus=1

If section\_size Reaches section\_capacity then

Update Schedule\_Status

If class\_status =0 and Waitinglist\_size<.specified size then

Update waitingList position of student

If waiting list is full then

Dbms\_output.put\_line(‘full can’t enter student’)

**Feature 13:** Allow a student to drop a course.

Input: Input includes student id and schedule id.

Output: First check whether the student has registered for that scheduled section (the status could be enrolled or wait listed). If not, print a message the student is not registered with that course.

Otherwise if the student is on wait list, remove the student, change registration status to dropped, and move up anyone after the student on wait list.

Exec Drop\_Course(Schedule\_id,Student\_id);

**Feature 14:** Allow a student to print a course schedule for a given year and semester

Input: Input includes student id, year, and semester

Output: Print course id, course name, section id, and status of all courses the student has registered in that year and semester. For any course the student is on the waiting list, also print out the waiting list position.

Exec dbms\_output.put\_line(student\_id || Year || Semester);

**Feature 15:** Print enrollment statistics for a department given department id, year, and semester.

Input: Input includes department id, year, and semester

Output: a. Print out total number of students enrolled for at least one course (do not count dropped or wait listed cases) in any courses in that year and semester in programs in the department.

Exec dbms\_output.put\_line(Total\_student);

Output:b. Print out total number of courses in that department, year and semester, total number of course sections, #of students enrolled and wait listed in each course section along with course id, course name, section id.

Exec dbms\_output.put\_line(Total\_courses || Year || Semester || Total\_course\_section || Students\_enrolled || Wait\_list || Course\_id || Course\_name || Section\_id);

**Feature 16:** Identify for a given year and semester, the top k courses with the longest waiting list, the top k rooms with the fewest scheduled class sections, and the top k time blocks that have the fewest scheduled class sections.

Input: Input includes year, semester, and k.

Output: print out the id and name of classes, their waiting list length. For rooms, please print out room id, room name, and #of scheduled class sections. For time blocks print out time block id, days of the week, and start time.

Exec dbms\_output.put\_line(Class\_id || Class\_name || Waitlist\_time);

Exec dbms\_output.put\_line(room\_id || room\_name || number\_schedule\_class\_sections);

Exec dbms\_output.put\_line(block\_id || Week\_days || Start\_time);