**Functions**

//this function creates a list of timetables for every class of a department.

GenerateDepartmentTimetable(class[], Rooms) {

totalClasses = lengthOf(class)

TimeSlot timetable[totalClasses]

x = 0

Stack FreeRooms, BusyRooms

for i = 0 to totalClasses

for x to class[i].noOfSections{

FreeRooms.push(Room[x]) //this ensures that the first few classes aren’t prioritized and are arranged in a minimum no of classes

x = class[i].noOfSections + 1

timetable[i] = GenerateClassTimetable(class[i], FreeRooms, BusyRooms)

}

GenerateClassTimetable(class, FreeRooms, BusyRooms) {

totalNoOfCourses = len(class.courses[])

x = 0

TimeSlot timetable[5, class.dailyWorkingHours, class.noOfSections]

TimeSlot FreeSlots[]

y = 0

Counter counter(noOfSections, totalCourses)

curCourse = class.courses[x]

for i = 1 to 5 {

for j=1 to class.dailyWorkingHours {

classesUsed = 0

for k = 1 to class.noOfSections {

while timetable[i][j][k]<0 {

if (fit(curCourse, FreeRooms, i, j, k, timetable, counter) > 3 {

r = FreeRooms.pop()

timetable[i][j][k](curCourse, curCourse.teacher, r) //We are assigning the timetable the first found optimal value

BusyRooms.push(r)

classesUsed = classesUsed+1

}

else {

if (x>totalNoOfCourses) {

x = 0

curCourse = courses[x]

}

else {

x++

curCourse = courses[x]

}

}

}

if timetable[i][j][k]=0 {

FreeSlots[y] = (i,j,k)

y++

}

if (x>totalNoOfCourses) {

x = 0

curCourse = courses[x]

}

else {

x++

curCourse = courses[x]

}

}

while classesUsed != 0 {

r = B.pop()

F.push(r)

classesUsed = classesUsed-1

}

}

}

errorCourse[], errorSec[] = checkForErrors(counter, courses, totalNoOfCourses, noOfSections)

errorNum = lengthOf(errors)

if m == 0 {

return timetable

else

repairTimetable(timetable, FreeRooms, FreeSlots, counter, errorSec, errorCourse, errorNum, 0)

return timetable

}

fit (course, F, i, j, k, timetable, counter) {

var=0

if (F.peep!=-1)

var = var+1

if (TeacherIsAvailable(timetable, i, j, course.teacher)

var = var + 1

if (course.crtHours > counter.get(course, k))

var = var + 1

if (timetable[i][j-2][k] != course)

var = var+1

return var

}

checkForErrors(counter, course[], totalCourses, noOfSections) {

x = 0

for i=1 to totalCourses {

for j=1 to noOfSections {

if counter.section[i][j] != course[i].crtHours

errorCourse[x] = course[i] //store course index and section index to spot the problem

errorSec[x] = j

x++;

}

}

}

return errorCourse[] and errorSec[]

}

repairTimetable (timetable[][][], FreeRooms, FreeSlots, counter, errorSec[], errorCourse[], errorNo, x) {

if (errorSec[] is empty) {

return //also congrats

}

else {

curCourse = errorCourse[errorNo]

curSection = errorSec[errorNo]

while (errorCourse[errorNo]!= null) {

if fit (curCourse, FreeRooms, i, j, curSection, timetable, counter) > 3 {

i, j = FreeSlots[x]

x++

r = FreeRooms.pop()

timetable[i][j][curSection](curCourse, curCourse.teacher, r)

FreeRooms.push(r)

errorSec[errorNo] = null

errorCourse[errorNo] = null

} else {

if x == lengthOf(FreeSlots){

x=0

i, j = FreeSlots[x]

x++

} else {

i, j = FreeSlots[x]

x++

}

}

}

repairTimetable(timetable[][][], FreeRooms, FreeSlots, counter, errorSec[], errorCourse[], errorNo-1, x)

}

}