```
    var advisoryScheduleTimes = [["8:00","9:20"],["9:35","10:55"],["11:00","12:20"],["13:40","15:00"]]
    var regularScheduleTimes = [["8:00","9:25"],["9:40","11:05"],["11:15","12:40"],["13:35","15:00"]]
    var earlyReleaseTimes = [["8:00","8:55"],["9:00","9:55"],["10:20","11:15"],["11:20","12:15"]]
    var extendedHourTimes = [["13:40","15:30"],["13:10","15:00"]]


    var calendarURL = ""
    var calendar;



function GenerateMySchedule(){
  const extendedStartDate = new Date("August 28 2019")
  setUserCalendar()
  var myBlocks = getClassBlocksFromSheet()
  var calendarData = getCalendarInfoFromSheet()

  for each (var dayData in calendarData){
    for each (var block in myBlocks){

    if(isTodayASchoolDay(dayData[0]) && checkIfBlockIsInCellString(dayData[0], block[0])){


      var blockTimes = getTimesForBlockFromCellString(dayData[0], dayData[1], block[0], block[1])
      Logger.log(blockTimes)
      var hasExtendedHours = checkIfExtendedHoursShouldHappenToday(block[0], getScheduleOrderFromString(dayData[0]),checkIfCourseHasExtendedHours(block[1]))

      var currentDate = new Date(dayData[1])
      if(currentDate > extendedStartDate){
        createCalendarEventForBlock(block[0],block[1], dayData[1], blockTimes[0], blockTimes[1], hasExtendedHours)
      }
      else{
        createCalendarEventForBlock(block[0],block[1], dayData[1], blockTimes[0], blockTimes[1], false)
      }
      Utilities.sleep(150);
       }
  }
      }
  var ui = SpreadsheetApp.getUi();
  var response = ui.alert('Process complete. Check your calendar to confirm that the events were added correctly')

}
```

```
function getClassBlocksFromSheet(){

  var myBlocks = []

  var sheet = SpreadsheetApp.getActiveSheet();
  var classBlocks = sheet.getRange("ScheduleInformation!B5").getDataRegion(SpreadsheetApp.Dimension.ROWS).getDisplayValues()
  var classNames = sheet.getRange("ScheduleInformation!C5").getDataRegion(SpreadsheetApp.Dimension.ROWS).getDisplayValues()

  var numOfClasses = classBlocks.length - 1;
  for(var i = 1;i<=numOfClasses;i++){
    myBlocks.push([classBlocks[i][0],classNames[i][0]])
  }
  //Logger.log(myBlocks)
 return myBlocks;
}


function getCalendarInfoFromSheet(){

  var calendarTable = []

var sheet = SpreadsheetApp.getActiveSheet();
var dayDescriptionString = sheet.getRange("DayCalendar!B2").getDataRegion(SpreadsheetApp.Dimension.ROWS).getDisplayValues()
var dateString = sheet.getRange("DayCalendar!C2").getDataRegion(SpreadsheetApp.Dimension.ROWS).getDisplayValues()

var numOfDays = dayDescriptionString.length - 1;
  for(var i = 1;i<=numOfDays;i++){
    calendarTable.push([dayDescriptionString[i][0],dateString[i][0]])
  }
  return calendarTable;
}
```

```
function setUserCalendar(){

var sheet = SpreadsheetApp.getActiveSheet();
calendarURL = sheet.getRange("ScheduleInformation!B2").getDisplayValue()
calendar = CalendarApp.getCalendarById(calendarURL);
}

function clearCalendarOfEvents(){

  var sheet = SpreadsheetApp.getActiveSheet();
calendarURL = sheet.getRange("ScheduleInformation!B2").getDisplayValue()
calendar = CalendarApp.getCalendarById(calendarURL);
  var fromDate = new Date("August 6 2019");
    var toDate = new Date("May 28 2020");
    var calendarName = 'My Calendar';


    var events = calendar.getEvents(fromDate, toDate);

    for(var i=0; i<events.length;i++){
      var ev = events[i];
      if(ev.getDescription() == "This event was added by a robot."){
      ev.deleteEvent();
      };
      Utilities.sleep(100);
    }
  var ui = SpreadsheetApp.getUi();
  var response = ui.alert('Process complete. Check your calendar to confirm that all blocks have been removed.')


}
function checkIfBlockIsInCellString(string, block){
  //This is calling another function called getScheduleOrderFromString.
  var scheduleOrder = getScheduleOrderFromString(string)
  if(scheduleOrder.indexOf(block) == -1){
    return false;
  }
  return true

}

function isTodayASchoolDay(string){
  //This is another regular expression that checks to see if a string contains 'Day' followed by a number between 1 and 8.
  normalDayRegEx = new RegExp(/Day [1-8]/i)

  if (normalDayRegEx.test(string)){return true}

  return false

}
```

```
function getTimesForBlock(blockName,scheduleOrder, scheduleType, hasExtendedHours){

  var scheduleIndex = scheduleOrder.indexOf(blockName)

  //A switch statement runs code depending on the value inside of it.
  //Here, the variable value scheduleType decides which code will run.
  switch (scheduleType){
    case "Adv":

      //Notice what happens here. If extended hours should happen today is true, the times used are taken from the extendedHourTimes array.
      if(checkIfExtendedHoursShouldHappenToday(blockName, scheduleOrder, hasExtendedHours)){
     //Logger.log("extended hours apply adv")
        return extendedHourTimes[0];

      }
      else{
        //Logger.log("adv hours apply")
      return advisoryScheduleTimes[scheduleIndex]
      }
      break;

    case "Early":
      //Logger.log("early hours apply")
      return earlyReleaseTimes[scheduleIndex]
      break;

    default:
      if(checkIfExtendedHoursShouldHappenToday(blockName, scheduleOrder, hasExtendedHours)){
        //Logger.log("extended hours apply reg")
        return extendedHourTimes[1];

      }
      else{
        //Logger.log("reg hours apply")
      return regularScheduleTimes[scheduleIndex]
      }
      break;
  }


}
```

```
function getScheduleOrderFromString(string){

  //The string is always the contents of the descriptive cell of the spreadsheet.
  //The split command breaks a string into pieces wherever the string inside occurs. These pieces are stored in an array.
 var dayNumber = string.split("Day ")[1].split(" ")[0]

 switch(parseInt(dayNumber)){

   case 1:
     return "ABCD"
     break
   case 2:
   return "EFGH"
     break
   case 3:
   return "BCDA"
     break
   case 4:
   return "FGHE"
     break
   case 5:
   return "CDAB"
     break
   case 6:
   return "GHEF"
     break
   case 7:
   return "DABC"
     break
   case 8:
   return "HEFG"
     break

     default:
     return "error"

 }

}
```

```
function getTimesForBlockFromCellString(string,date,block,description){

  const extendedStartDate = new Date("August 28 2019") //Extended hours start after August 28th.
  const currentDate = new Date(date)


  var scheduleType = getScheduleTypeFromCellString(string)
  var scheduleOrder = getScheduleOrderFromString(string)

  var hasExtendedHours = checkIfCourseHasExtendedHours(description) && (currentDate > extendedStartDate)

  return getTimesForBlock(block,scheduleOrder,scheduleType,hasExtendedHours)

}
```

```
function checkIfExtendedHoursShouldHappenToday(blockName, scheduleOrder, hasExtendedHours){

  //The .indexOf command of a string determines the first character at which a particular string or character occurs.
  //For example, "ZYXWVUTSRQ".indexOf('U') has a value of 5.
  //Another example: "FOOTLOOSE".indexOf('O') has a value of 1.
  //Even though there are multiple 'O's in the name, the indexOf command gives the index of the first.

  var scheduleIndex = scheduleOrder.indexOf(blockName)
  var courseBlockOccursAtEndOfDay = (scheduleIndex == 3)

  return (hasExtendedHours && courseBlockOccursAtEndOfDay)

}

function checkIfCourseHasExtendedHours(courseDescription){

  //A regular expression is a way to search text for a specific pattern.
   var APRegEx = new RegExp(/AP/)
   var IBRegEx = new RegExp(/HL/)

   return APRegEx.test(courseDescription) | IBRegEx.test(courseDescription)

}
```

```
function stringForPrintTimes(times){

  //This returns a string that shows the times for the block in a nice way.
  //times[0] means the first element in the times array.
  //You can add strings together using the + symbol.

  return times[0] + " - " + times[1];
}

function getScheduleTypeFromCellString(string){

  //A regular expression is contained within two forward slashes as shown below.
  //The /i means that the pattern should be found whether it occurs with capital or lowercase letters.

  advRegEx = new RegExp(/adv/i)
  earlyRegEx = new RegExp(/early/i)

  //If you have a regular expression object (created using RegExp), the RegExp.test command checks whether or not a pattern appears.
  //For example, advRegEx.test("I advise you to sleep now") will be a true statement because the string contains the letters 'adv' in a row.
  //Another example: advRegEx.test("Early in the morning they arose") also is a true statement.

  //A counter example: earlyRegEx.test("I saw your ear lying on the ground") will be false. The space between the 'ear' and 'ly' does not match the string.

  if (advRegEx.test(string)){return "Adv"}
  if(earlyRegEx.test(string)){return "Early"}
  return "Reg"

}
function createCalendarEventForBlock(blockName, className, classDate, startTime, endTime, isExtended){


  if(isExtended){
    var extendedString = " (Extended Block)"
  }
  else{
    var extendedString = ""
  }

  var eventTitle = "(" + blockName + ") " + className + extendedString
  var startDateTime = new Date(classDate + " " + startTime)
  var endDateTime = new Date(classDate + " " + endTime)

  calendar.createEvent(eventTitle, startDateTime, endDateTime, {description: 'This event was added by a robot.'})


}
```