# Task Overview:

You are to write two short programs, as described below.  
**WHEN YOU ARE FINISHED** you are to create a Google doc and answer the questions on the final page of this document.

For each program, a sample run will be shown. Your program should match this sample AS CLOSELY AS POSSIBLE. In the example, any **underlined text** represents information which was typed in on the keyboard and did NOT come from a print statement.

# First program:

You have been tasked with creating a dice roller application that simulates different kinds of dice of varying number of sides. You are to write a program in a class called **DiceRoller**

Your program will begin by printing a menu for the user to make a selection from. The choices should be:

* 4-sided die (generates numbers 1 to 4)
* 6-sided die (generates numbers 1 to 6)
* 8-sided die (generates numbers 1 to 8)
* 10-sided die (generates numbers 1 to 10)
* 20-sided die (generates numbers 1 to 20)

The user should type in the number of sides on the die they would like to roll. If they type in a number that is not in the list, they should receive an error message and the program will end.

Afterwards, your program will **generate a random integer** in the correct range, and then tell the user what they rolled.

If the user had chosen to roll the 20-sided die, it should also output a message based on what number was rolled:

* If a 1 was rolled, it should print out “VERY BAD”.
* If a number from 2 to 9 was rolled, it should print out “Poor roll”.
* If a number from 10 to 14 was rolled, it should print out “Average roll”.
* If a number from 15 to 19 was rolled, it should print out “Good roll”.
* If a 20 was rolled, it should print out “VERY GOOD”

Any variables used should be of an **appropriate data type**. Any control structures used should be the best choice for the situation they are used in.

Your program should follow **good coding practices** and must have a **program header block** and in-line comments.

Sample output follows on the next page.  
It shows the same program being run 3 different times.

# Sample Output, run #1:

========== ELECTRONIC DICE ROLLER ==========

Please enter the number for the kind of die you would like to roll:

4-sided die (generates numbers from 1 to 4)

6-sided die (generates numbers from 1 to 6)

8-sided die (generates numbers from 1 to 8)

10-sided die (generates numbers from 1 to 10)

20-sided die (generates numbers from 1 to 20)

What kind of die do you want to roll? **4**

You rolled a: 3

# Sample Output, run #2:

========== ELECTRONIC DICE ROLLER ==========

Please enter the number for the kind of die you would like to roll:

4-sided die (generates numbers from 1 to 4)

6-sided die (generates numbers from 1 to 6)

8-sided die (generates numbers from 1 to 8)

10-sided die (generates numbers from 1 to 10)

20-sided die (generates numbers from 1 to 20)

What kind of die do you want to roll? **20**

You rolled a: 1

VERY BAD

# Sample Output, run #3:

========== ELECTRONIC DICE ROLLER ==========

Please enter the number for the kind of die you would like to roll:

4-sided die (generates numbers from 1 to 4)

6-sided die (generates numbers from 1 to 6)

8-sided die (generates numbers from 1 to 8)

10-sided die (generates numbers from 1 to 10)

20-sided die (generates numbers from 1 to 20)

What kind of die do you want to roll? **20**

You rolled a: 17

Good rollSecond program:

You have been tasked with writing an application to help keep track of the parking spots in parking lot. You are to write a program in a class called **ParkingTracker**

The parking lot has 12 parking spots – **three rows** with **four parking spots** in each row.

Your program will use a **multi-dimensional** array to track the positions in the parking lot.

How you choose to represent the parking lot in the array is up to you. (For example, you could choose to create an array of int, where 0 is an empty spot, and 1 is a filled spot, or you could choose to use an array of chars using ‘X’ for a filled spot or a space for an empty spot. It would also be possible to use an array of boolean values. Do what you feel makes the most sense to you.)

The program will begin by **printing the empty parking** lot on screen and how many spots are available. Your program should then ask the user to **choose the row** and **column** numbers of a spot in the parking lot. If the **spot was empty**, it should **become filled**. If the **spot was filled**, it should **become empty**.

Your program should then loop to allow the user to continue typing in more parking spots. Each time they do so, the spot should be updated as above and the parking lot will be reprinted.

If the user ever types in a parking spot that does not exist (by entering an incorrect row or column number) the **program should end**.

You should **print a box around the parking** lot to show how big it is, and the row and column numbers should be shown.

When printing the parking lot, an **X** should be printed in any **spot that is full**, and a **space** should be printed in any **spot that is empty**.

Any variables used should be of an **appropriate data type**. Any control structures used should be the best choice for the situation they are used in. Constants should be used where appropriate.

Your program should follow **good coding practices** and must have a **program header block** and in-line comments.

**If you are unable to solve this problem using a 2D array:** You may do so using a single-dimensional array for fewer marks. The array should have 12 positions, and you may print out your parking lot in a single line instead of as a grid.

Sample output follows on the next page.

# 

# Sample Output:

0123

+----+

0| |

1| |

2| |

+----+

There are 12 spots available.  
Please choose a row: **1**

Please choose a column: **2**

Spot 1-2 filled!

0123

+----+

0| |

1| X |

2| |

+----+

There are 11 spots available.  
Please choose a row: **1**

Please choose a column: **1**

Spot 1-1 filled!

0123

+----+

0| |

1| XX |

2| |

+----+

There are 10 spots available.  
Please choose a row: **1**

Please choose a column: **2**

Spot 1-2 emptied!

0123

+----+

0| |

1| X |

2| |

+----+

There are 11 spots available.  
Please choose a row: **5**

Please choose a column: **2**

Invalid spot! Program ending.

# Google Doc Questions:

Create a Google Doc to answer the following questions. When completed, save it as a .pdf and submit it along with your program when you turn in your evaluation.

1. **If you used an if-block** in one of your programs:  
   Give the name of the program and the **beginning AND ending line number** of an if-block you used. Explain why you chose to use an if-block here instead of using a **switch statement**.  
   Would a switch statement have worked here? Explain why, or why not.

I used an if statement in the DiceRoller program from line 57-67, I used an if statement to check what to output for the 20-sided die. I used a if-block instead of a switch statment because a switch statement would be a lot less efficient as you would have to write every number within a range.

1. **If you used a switch-statement** in one of your programs:  
   Give the name of the program and the **beginning AND ending line number** of a switch-statement you used. Explain why you chose to use a switch-statement here instead of using an if-statement.

I used a switch statement in my DiceRoller program from 35-72 because it is 5 different options of specific numbers so I chose to use a switch statement instead. It was to select how many sides of a die and was a fixed value in which it is more effective in using a switch statement.  
  
**If you did NOT use a switch-statement** in your programs:

* 1. Explain the situations in which a switch-statement is better to use in a program, and the situations in which an if-block is better to use.
  2. Explain why you did not need to include switch statements in your programs.

1. **If you used a while-loop** in one of your programs:  
   Give the name of the program and the **beginning AND ending line number** of a while-loop you used. Explain why you chose to use a while loop here instead of using a **do-loop** or a **for-loop**.  
     
   **If you did NOT use a while-loop** in your programs, explain the situations in which a while-loop is the best loop type to use in a program, and why you did not need to use a while-loop in your programs.

The best situation for a while loop is when I need a code to be executed repeatedly until a condition is met. I did not use a while loop because I needed the code to at least run one time and it was ineffective if it could not be executed at all.

1. **If you used a do-loop** in one of your programs:  
   Give the name of the program and the **beginning AND ending line number** of a do-loop you used. Explain why you chose to use a do-loop here instead of using a **while-loop** or a **for-loop**.

I used a do-while loop in the ParkingTracker program from line 27-76

I used a do-while loop because I needed the code to at least execute once in order to get user input and print out the parking lot before any changes were made. And I can’t use a for loop because I need the program to run repeatedly.

**If you did NOT use a do-loop** in your programs, explain the situations in which a do-loop is the best loop type to use in a program, and why you did not need to use a do-loop in your programs.  
  
  
**QUESTIONS CONTINUE ON NEXT PAGE**

1. **If you used a for-loop** in one of your programs:  
   Give the name of the program and the **beginning AND ending line number** of a for-loop you used. Explain why you chose to use a for-loop here instead of using a **while-loop** or a **do-loop**.  
     
   **If you did NOT use a for-loop** in your programs, explain the situations in which a for-loop is the best loop type to use in a program, and why you did not need to use a for-loop in your programs.

The best situation for a for-loop is when you need a loop to run a specific amount of time howeveri n my program I needed them to run definitely until a condition is met.