

JS javascript 4-5 assignment

Practice exercise 4.1

1. Create a variable with a Boolean value.
2. Output the value of the variable to the console.
3. Check whether the variable is true and if so, output a message to the console.

Solution:

```
// 1. Create a variable with a Boolean value
```

```
let isAvailable = true;
```

```
// 2. Output the value of the variable to the console
```

```
console.log("The value of isAvailable is:", isAvailable);
```

```
// 3. Check whether the variable is true and if so, output a message to the console
```

```
if (isAvailable) {  
    console.log("The variable is true!");  
}
```

Output:

```
The value of isAvailable is: true  
The variable is true!
```

Practice exercise 4.2

1. Create a prompt to ask the user's age
2. Convert the response from the prompt to a number

3. Declare a message variable that you will use to hold the console message for the user
4. If the input age is equal to or greater than 21, set the message variable to confirm entry to a venue and the ability to purchase alcohol
5. If the input age is equal to or greater than 19, set the message variable to confirm entry to the venue but deny the purchase of alcohol
6. Provide a default else statement to set the message variable to deny entry if none are true
7. Output the response message variable to the console

Solution:

```
// 1. Create a prompt to ask the user's age
```

```
let userAge = prompt("Please enter your age:");
```

```
// 2. Convert the response from the prompt to a number
```

```
userAge = Number(userAge);
```

```
// 3. Declare a message variable that you will use to hold the console message for the user
```

```
let message;
```

```
// 4. If the input age is equal to or greater than 21
```

```
if (userAge >= 21) {
```

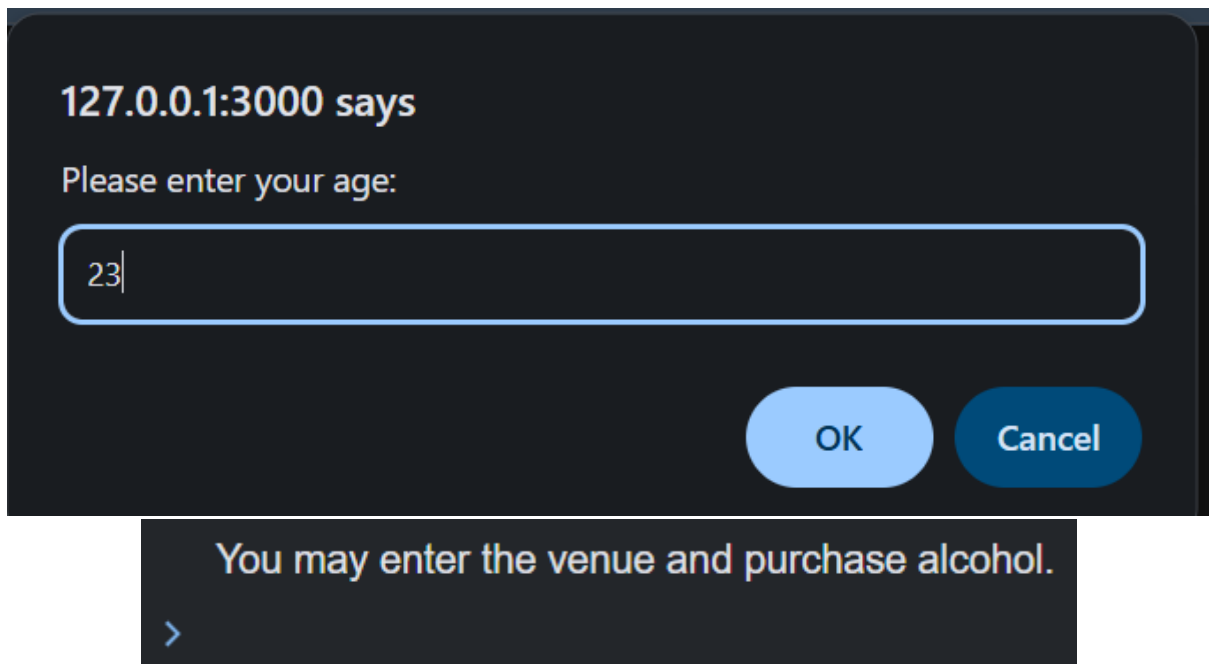
```
    message = "You may enter the venue and purchase alcohol.";
```

```
}
```

```
// 5. If the input age is equal to or greater than 19
```

```
else if (userAge >= 19) {  
    message = "You may enter the venue, but you may not purchase alcohol.";  
}  
// 6. Provide a default else statement  
else {  
    message = "You are not allowed to enter the venue.";  
}  
  
// 7. Output the response message variable to the console  
console.log(message);
```

Output:



Practice exercise 4.3

1. Create a Boolean value for an ID variable

2. Using a ternary operator, create a message variable that will check whether their ID is valid and either allow a person into a venue or not
3. Output the response to the console

Solution:

```
// 1. Create a Boolean value for an ID variable
```

```
let isValidID = true; // You can change this to false to test the other condition
```

```
// 2. Using a ternary operator, create a message variable that will check  
whether their ID is valid
```

```
let message = isValidID ? "Access granted: You may enter the venue." :  
"Access denied: Valid ID required.";
```

```
// 3. Output the response to the console
```

```
console.log(message);
```

Output:

```
Access granted: You may enter the venue.
```

Practice exercise 4.4

the JavaScript function `Math.random()` will return a random number in the range of 0 to less than 1, including 0 but not 1. You can then scale it to the desired range by multiplying the result and using `Math.floor()` to round it down to the nearest whole number;

Solution:

```
let randomNumber = Math.floor(Math.random() * 10) + 1;
```

```
console.log("Random number (1-10):", randomNumber);
```

Output:

Random number (1-10): 8

Practice exercise 4.5

1. Create a variable called prize and use a prompt to ask the user to set the value by selecting a number between 0 and 10
2. Convert the prompt response to a number data type
3. Create a variable to use for the output message containing the value "My Selection: "
4. Using the switch statement (and creativity), provide a response back regarding a prize that is awarded depending on what number is selected
5. Use the switch break to add combined results for prizes
6. Output the message back to the user by concatenating your prize variable strings and the output message string

Solution:

```
// 1. Create a variable called prize and use a prompt to ask the user to select a number between 0 and 10
```

```
let prize = prompt("Select a number between 0 and 10 to win a prize:");
```

```
// 2. Convert the prompt response to a number data type
```

```
prize = Number(prize);
```

```
// 3. Create a variable for the output message
```

```
let message = "My Selection: " + prize + " - ";
```

```
// 4 & 5. Use switch statement to determine prize
```

```
switch (prize) {
```

```
  case 0:
```

```
    message += "Oops! No prize this time. Try again!";
```

```
    break;
```

```
  case 1:
```

```
  case 2:
```

```
    message += "You won a chocolate bar!";
```

```
    break;
```

```
  case 3:
```

```
  case 4:
```

```
    message += "You won a movie ticket!";
```

```
    break;
```

```
  case 5:
```

```
  case 6:
```

```
    message += "You won a gift card worth $10!";
```

```
    break;
```

```
  case 7:
```

```
  case 8:
```

```
    message += "You won a Bluetooth speaker!";
```

```
    break;
```

```
  case 9:
```

```
  case 10:
```

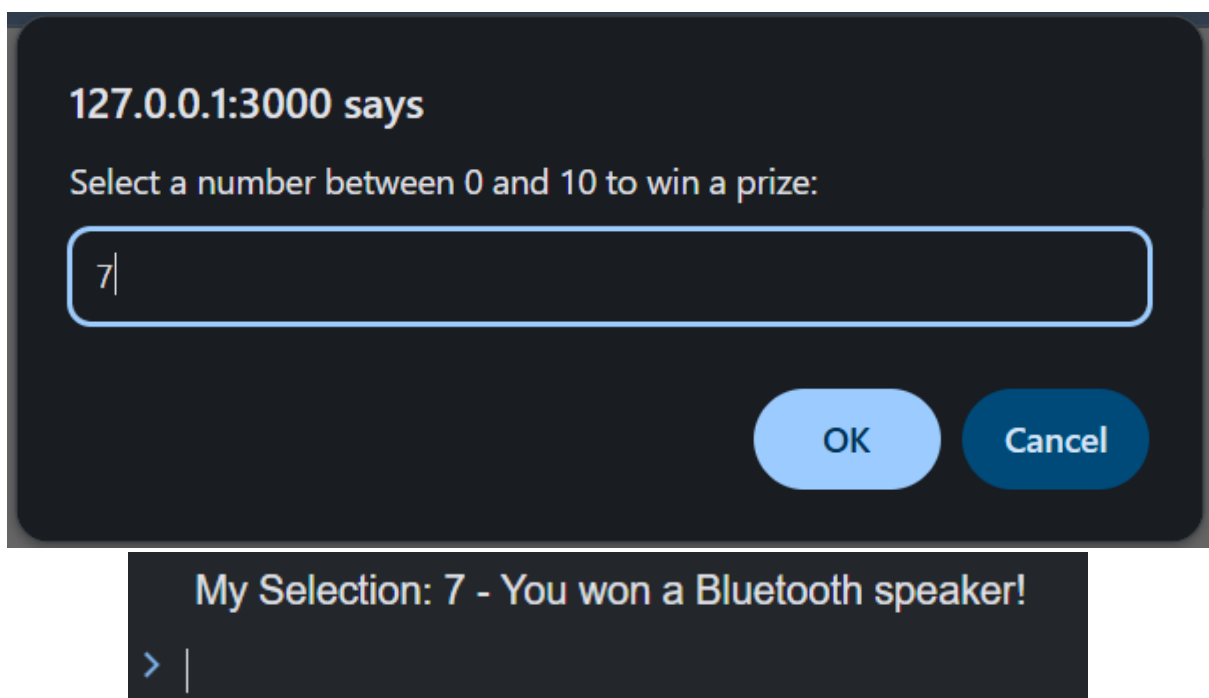
```
    message += "Congratulations! You won a brand new tablet!";
```

```
        break;
    default:
        message += "Invalid selection. Please pick a number between 0 and 10.";
        break;
}
```

// 6. Output the message back to the user

```
console.log(message);
```

Output:



Practice exercise 5.1

In this exercise we will create a number guessing game that takes user input and replies based on how accurate the user's guess was.

1. Create a variable to be used as the max value for the number guessing game.

2. Generate a random number for the solution using `Math.random()` and `Math.floor()`. You will also need to add 1 so that the value is returned as 1-[whatever the set max value is]. You can log this value to the console for development to see the value as you create the game, then when the game is complete you can comment out this console output.

Solution:

```
// 1. Create a variable to be used as the max value for the number guessing game
```

```
let maxValue = 10;
```

```
// 2. Generate a random number between 1 and maxValue using Math.random() and Math.floor()
```

```
let solution = Math.floor(Math.random() * maxValue) + 1;
```

```
// Log the solution to the console for development/testing
```

```
console.log("Secret number (for testing):", solution);
```

```
// Later, you can comment out the line above once the game is ready
```

Output:

```
Secret number (for testing): 10
```

Practice exercise 5.2

In this exercise, we will create a basic counter that will increase a dynamic variable by a consistent step value, up to an upper limit.

1. Set the starting counter to 0

2. Create a variable, step, to increase your counter by
3. Add a do while loop, printing the counter to the console and incrementing it by the step amount each loop
4. Continue to loop until the counter is equal to 100 or more than 100

Solution:

```
// 1. Set the starting counter to 0
```

```
let counter = 0;
```

```
// 2. Create a variable, step, to increase your counter by
```

```
let step = 10; // You can change this value as needed
```

```
// 3. Add a do while loop
```

```
do {
```

```
    console.log("Counter:", counter); // Print the counter
```

```
    counter += step; // Increment by step
```

```
} while (counter < 100); // 4. Continue until counter is equal to or more than 100
```

Output:

```
Counter: 0
Counter: 10
Counter: 20
Counter: 30
Counter: 40
Counter: 50
Counter: 60
Counter: 70
Counter: 80
Counter: 90
Counter: 0
Counter: 10
Counter: 20
Counter: 30
Counter: 40
Counter: 50
Counter: 60
Counter: 70
Counter: 80
Counter: 0
Counter: 10
Counter: 20
Counter: 30
Counter: 40
Counter: 50
Counter: 60
Counter: 0
Counter: 10
Counter: 20
Counter: 30
Counter: 40
Counter: 0
Counter: 10
Counter: 20
Counter: 0
Counter: 0
Counter: 10
Counter: 0
Counter: 0
Counter: 10
Counter: 0
Counter: 0
Counter: 10
Counter: 20
Counter: 30
Counter: 40
Counter: 50
Counter: 60
Counter: 70
Counter: 80
Counter: 90
```

Practice exercise 5.3

In this exercise we will use a for loop to create an array that holds objects. Starting with creating a blank array, the block of code within the loop will create an object that gets inserted into the array.

1. Setup a blank array, myWork.
2. Using a for loop, create a list of 10 objects, each of which is a numbered lesson (e.g. Lesson 1, Lesson 2, Lesson 3....) with an alternating true/false status for every other item to indicate whether the class will be running this year. For example: name: 'Lesson 1', status: true
3. You can specify the status by using a ternary operator that checks whether the modulo of the given lesson value is equal to zero and by setting up a Boolean value to alternate the values each iteration.
4. Create a lesson using a temporary object variable, containing the name (lesson with the numeric value) and predefined status (which we set up in the previous step).
5. Push the objects to the myWork array. 6. Output the array to the console.

Solution:

```
// 1. Setup a blank array
```

```
let myWork = [];
```

```
// 2-5. Use a for loop to create and push lesson objects with alternating status
```

```
for (let i = 1; i <= 10; i++) {
```

```
    // 3. Use ternary operator to alternate status: true for odd lessons, false for even
```

```
    let status = i % 2 === 0 ? false : true;
```

```
// 4. Create a temporary object with lesson name and status
let lesson = {
  name: `Lesson ${i}`,
  status: status
};

// 5. Push the object to the array
myWork.push(lesson);
}

// 6. Output the array to the console
console.log(myWork);
```

Output:

```
[
  { name: 'Lesson 1', status: true },
  { name: 'Lesson 2', status: false },
  { name: 'Lesson 3', status: true },
  { name: 'Lesson 4', status: false },
  { name: 'Lesson 5', status: true },
  { name: 'Lesson 6', status: false },
  { name: 'Lesson 7', status: true },
  { name: 'Lesson 8', status: false },
  { name: 'Lesson 9', status: true },
  { name: 'Lesson 10', status: false }
]
```

Practice exercise 5.4

In this exercise we will be generating a table of values. We will be using loops to generate rows and also columns, which will be nested within the rows. Nested arrays can be used to represent rows in a table. This is a common structure in spreadsheets, where each row is a nested array within a table and the contents of these rows are the cells in the table. The columns will align as we are creating an equal number of cells in each row.

1. To create a table generator, first create an empty array, `myTable`, to hold your table data.
2. Set variable values for the number of rows and columns. This will allow us to dynamically control how many rows and columns we want within the table. Separating the values from the main code helps make updates to the dimensions easier.
3. Set up a counter variable with an initial value of 0. The counter will be used to set the content and count the values of the cells within the table.
4. Create a for loop with conditions to set the number of iterations, and to construct each row of the table. Within it, set up a new temporary array (`tempTable`) to hold each row of data. The columns will be nested within the rows, generating each cell needed for the column.
5. Nest a second loop within the first to count the columns. Columns are run within the row loop so that we have a uniform number of columns within the table.

Solution:

```
// 1. Create an empty array to hold the table data
```

```
let myTable = [];
```

```
// 2. Set variable values for rows and columns
let numRows = 4;
let numCols = 5;

// 3. Set up a counter variable
let counter = 0;

// 4. Outer loop to generate rows
for (let i = 0; i < numRows; i++) {
    // Create a temporary array for the current row
    let tempTable = [];

    // 5. Inner loop to generate columns (cells in a row)
    for (let j = 0; j < numCols; j++) {
        tempTable.push(counter); // Push the counter value to the current cell
        counter++; // Increment the counter
    }

    // Push the completed row (tempTable) to the main table (myTable)
    myTable.push(tempTable);
}

// Output the table
console.log(myTable);
```

Output:

```
[  
  [ 0, 1, 2, 3, 4 ],  
  [ 5, 6, 7, 8, 9 ],  
  [ 10, 11, 12, 13, 14 ],  
  [ 15, 16, 17, 18, 19 ]  
]
```

Practice exercise 5.5

Explore how to create a table grid that contains nested arrays as rows within a table. The rows will each contain the number of cells needed for the number of columns set in the variables. This grid table will dynamically adjust depending on the values for the variables.

1. Create a grid array variable.
2. Set a value of 64 for the number of cells.
3. Set a counter to 0.

Solution:

```
// 1. Create a grid array variable
```

```
let grid = [];
```

```
// 2. Set a value of 64 for the number of cells (total cells = rows * columns)
```

```
let totalCells = 64;
```

```
// We will use 8 rows and 8 columns for this example
```

```
let numRows = 8;
```

```
let numCols = 8;
```

```
// 3. Set a counter to 0
```

```
let counter = 0;
```

```
// Create the grid with nested arrays (rows)
```

```
for (let i = 0; i < numRows; i++) {
```

```
    // Create a temporary array for the current row
```

```
    let row = [];
```

```
    // Add cells to the row
```

```
    for (let j = 0; j < numCols; j++) {
```

```
        row.push(counter); // Add the counter value to the current cell
```

```
        counter++; // Increment the counter
```

```
    }
```

```
    // Push the row into the grid
```

```
    grid.push(row);
```

```
}
```

```
// Output the grid
```

```
console.log(grid);
```


Output:

```
[  
  [  
    0, 1, 2, 3,  
    4, 5, 6, 7  
  ],  
  [  
    8, 9, 10, 11,  
    12, 13, 14, 15  
  ],  
  [  
    16, 17, 18, 19,  
    20, 21, 22, 23  
  ],  
  [  
    24, 25, 26, 27,  
    28, 29, 30, 31  
  ],  
  [  
    32, 33, 34, 35,  
    36, 37, 38, 39  
  ],  
  [  
    40, 41, 42, 43,  
    44, 45, 46, 47  
  ],  
  [  
    48, 49, 50, 51,  
    52, 53, 54, 55  
  ],  
  [  
    56, 57, 58, 59,  
    60, 61, 62, 63  
  ]  
]
```

Practice exercise 5.6

This exercise will construct an array as it loops through the incrementing values of x. Once the array is done, this exercise also will demonstrate several ways to output array contents.

1. Create an empty array
2. Run a loop 10 times, adding a new incrementing value to the array
3. Log the array into the console
4. Use the for loop to iterate through the array (adjust the number of iterations to however many values are in your array) and output into the console
5. Use the for of loop to output the value into the console from the array

Solution:

```
// 1. Create an empty array
```

```
let myArray = [];
```

```
// 2. Run a loop 10 times, adding a new incrementing value to the array
```

```
for (let x = 0; x < 10; x++) {
```

```
    myArray.push(x); // Add the incrementing value to the array
```

```
}
```

```
// 3. Log the array into the console
```

```
console.log("Array after loop:", myArray);
```

```
// 4. Use a for loop to iterate through the array and output the values
```

```
console.log("Using a for loop to iterate:");  
for (let i = 0; i < myArray.length; i++) {  
    console.log(myArray[i]); // Output each element of the array  
}
```

// 5. Use the for...of loop to output the value into the console

```
console.log("Using a for...of loop to iterate:");  
for (let value of myArray) {  
    console.log(value); // Output each value using for...of loop  
}
```

Output:

```
Array after loop: [  
  0, 1, 2, 3, 4,  
  5, 6, 7, 8, 9  
]  
Using a for loop to iterate:  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
Using a for...of loop to iterate:  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

Practice exercise 5.7

In this exercise, we will experiment with looping over objects and internal arrays.

1. Create a simple object with three items in it.
2. Using the for in loop, get the properties' names and values from the object and output them into the console.

3. Create an array containing the same three items. Using either the for loop or the for in loop, output the values from the array into the console.

Solution:

```
// 1. Create a simple object with three items
```

```
let myObject = {  
  name: "Alice",  
  age: 25,  
  occupation: "Engineer"  
};
```

```
// 2. Use a for in loop to get the properties' names and values from the object
```

```
console.log("Object Properties:");  
for (let key in myObject) {  
  console.log(key + ": " + myObject[key]); // Output property name and value  
}
```

```
// 3. Create an array containing the same three items
```

```
let myArray = ["Alice", 25, "Engineer"];
```

```
// Use a for loop to output the values from the array
```

```
console.log("\nArray Values (using for loop):");  
for (let i = 0; i < myArray.length; i++) {  
  console.log(myArray[i]); // Output each item from the array  
}
```

```
// Alternatively, use a for in loop to output values from the array
console.log("\nArray Values (using for in loop):");
for (let index in myArray) {
    console.log(myArray[index]); // Output each item from the array using the
index
}
```

Output:

```
Object Properties:
name: Alice
age: 25
occupation: Engineer
```

```
Array Values (using for loop):
Alice
Alice
25
Engineer
```

```
Array Values (using for in loop):
Alice
25
Engineer
```

Practice exercise 5.8

This exercise will demonstrate how to create a string with all the digits as it loops through them. We can also set a value to skip by adding a condition that will use `continue`, skipping the matching condition. A second option is to do the same exercise and use the `break` keyword.

1. Set up a string variable to use as output.

2. Select a number to skip, and set that number as a variable.
3. Create a for loop that counts to 10.
4. Add a condition to check if the value of the looped variable is equal to the number that should be skipped.
5. If the number is to be skipped in the condition, continue to the next number.
6. As you iterate through the values, append the new count value to the end of the main output variable.
7. Output the main variable after the loop completes.
8. Reuse the code, but change the continue to break and see the difference. It should now stop at the skip value.

Solution:

```
// 1. Set up a string variable to use as output
```

```
let output = "";
```

```
// 2. Select a number to skip
```

```
let skipNumber = 5;
```

```
// 3. Create a for loop that counts to 10
```

```
console.log("Using continue to skip number:");
```

```
for (let i = 0; i < 10; i++) {
```

```
  // 4. Add a condition to check if the looped variable is equal to the number  
  to skip
```

```
    if (i === skipNumber) {
```

```
      // 5. If the number is to be skipped, continue to the next number
```

```
      continue;
```

```
}  
  
// 6. Append the new count value to the output string  
output += i;  
}  
  
// 7. Output the final string after the loop completes  
console.log(output); // Output will not include the skipped number (5)  
  
// Reset output for the next loop  
output = "";  
  
// 8. Reuse the code, but change the continue to break and see the difference  
console.log("\nUsing break to stop at the skip number:");  
  
for (let i = 0; i < 10; i++) {  
    // 4. Add a condition to check if the looped variable is equal to the number  
    // to skip  
    if (i === skipNumber) {  
        // 5. If the number is to be skipped, break out of the loop  
        break;  
    }  
    // 6. Append the new count value to the output string  
    output += i;  
}  
  
// 7. Output the final string after the loop completes  
console.log(output); // Output will stop at the skipped number (5)
```


Output:

```
Using continue to skip number:  
012346789
```

```
Using break to stop at the skip number:  
01234
```

Math multiplication table

In this project, you will create a math multiplication table using loops. You can do this using your own creativity or by following some of the following suggested steps:

1. Set up a blank array to contain the final multiplication table.
2. Set a value variable to specify how many values you want to multiply with one another and show the results for.
3. Create an outer for loop to iterate through each row and a temp array to store the row values. Each row will be an array of cells that will be nested into the final table.
4. Add an inner for loop for the column values, which will push the multiplied row and column values to the temp array.
5. Add the temporary row data that contains the calculated solutions to the main array of the final table. The final result will add a row of values for the calculations.

Solution:

```
// 1. Set up a blank array to contain the final multiplication table
```

```
let multiplicationTable = [];
```

```
// 2. Set a value variable to specify how many values you want to multiply
```

```
let values = 10; // Multiplying 1 to 10
```

```
// 3. Create an outer for loop to iterate through each row
```

```
for (let i = 1; i <= values; i++) {
```

```
    let tempRow = []; // Temporary array to store the current row's results
```

```
    // 4. Add an inner for loop for the column values
```

```
    for (let j = 1; j <= values; j++) {
```

```
        // Push the multiplied values to the temp array
```

```
        tempRow.push(i * j); // Multiply i (row) by j (column)
```

```
    }
```

```
// 5. Add the temporary row data to the main array of the final table
```

```
multiplicationTable.push(tempRow);
```

```
}
```

```
// Output the multiplication table
```

```
console.log("Multiplication Table:");
```

```
for (let row of multiplicationTable) {
```

```
    console.log(row.join('\t')); // Join each row's values with a tab for proper  
    spacing
```

```
}
```

Output:

Multiplication Table:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
2	4	6	8	10	12	14	16	18	20
2	4	6	8	10	12	14	16	18	20
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100