Mohan Sai Potla

24 Oct 2022

JavaScript Assignment 15

1). Create a function that finds the maximum range of a triangle's third edge, where the side lengths are all integers.

Examples

nextEdge(8, 10) ➞ 17

nextEdge(5, 7) ➞ 11

nextEdge(9, 2) ➞ 10

```
//function that finds the maximum range of a triangle's third edge, where the side lengths are all integers.
function nextEdge(side1, side2) {
    if(side1 > 0 && side2 > 0){
        return (side1 + side2) -1;
    }
    return (`Invalid Inputs`);
}

//driver code
let result = nextEdge(8,10);
console.log(result);
```

2). The **right shift** operation is similar to **floor division by powers of two**. Write a function that **mimics** (without the use of **>>**) the right shift operator and returns the result from the two given integers. Try to solve this challenge by recursion.

```
//function that mimics the right shift operator and returns the result from the two given integers.
```

```
function rightShift(num, den) {
    return Math.floor(num / Math.pow(2 , den));
}

//driver code
let result = rightShift(20,2);
console.log(result);
```

3). Create a function that takes numbers b and m as arguments and returns the second derivative of the function f(x)=x^b +x* (e^(b*m)) with respect to x evaluated at x=m, where b and m are constants.

```
//double derivative of given function is f''(x) = (b^2-b) * x ^(b-2)
// At x=m it becomes (b^2 -b) * m ^ (b-2)
function secondDerivative (b ,m) {
    let ans = (Math.pow(b,2) -b) * (Math.pow(m, (b-2)));

    return ans;
}

//driver code
let result = secondDerivative(4,2);
console.log(result);
```

4). This Triangular Number Sequence is generated from a pattern of dots that form a triangle. The first 5 numbers of the sequence, or dots, are:
1, 3, 6, 10, 15
This means that the first triangle has just one dot, the second one has three dots, the third one has 6 dots and so on.
Write a function that returns the cumulative sum of the number of all the previous (including current) dots when given its corresponding triangle number of the sequence.

```
//function that returns the cumulative sum of the number of all the previous (including current) dots when given its corresponding triangle number of the sequence
function dotsInTriangle(num){
    let sum =0,total=0;
    for(let i =1; i<=num; i++){
        sum += i;
        total += sum;
    }
    return total;
}

//driver code
let result = dotsInTriangle(4);
console.log(result);
```

5). Given a total due and an array representing the amount of change in your pocket, determine whether or not you are able to pay for the item. Change will always be represented in the following order: quarters, dimes, nickels, pennies.
To illustrate: changeEnough([25, 20, 5, 0], 4.25) should yield true, since having 25 quarters, 20 dimes, 5 nickels and 0 pennies gives you 6.25 + 2 + .25 + 0 = 8.50.

```
//function to determine whether you are able to pay for the item or not
function changeEnough(arr, itemPrice) {
    let totalMoney = arr[0]*0.25 + arr[1]*0.10 + arr[2]*0.05 + arr[3]*0.01;
    if(totalMoney >= itemPrice){
        return true;
    }else{
        return false;
    }
}
```

```
//driver code
let result = changeEnough([30, 40, 20, 5], 12.55);
console.log(result);
```