

Workshop 3

1. Area of a Triangle

Write a function that takes the base and height of a triangle and `return` its area.

Examples

`triArea(3, 2)` → 3

`triArea(7, 4)` → 14

`triArea(10, 10)` → 50

Notes

- The area of a triangle is: $(\text{base} * \text{height}) / 2$
- Don't forget to `return` the result.

work3q1.js × index.js × +

Index.js > ...

```
1 function AreaOfTriangle(base, height){  
2   var area=(base*height)/2;  
3   return area;  
4 }  
5  
6 let n1=parseInt(prompt("Enter the base"))  
7 let n2=parseInt(prompt("Enter the height"))  
8 console.log(AreaOfTriangle(n1,n2));  
9 |
```

Line 9 : Col 1

History ↻

_ Console × Shell × +

```
Enter the base> 5  
Enter the height> 6  
15  
Hint: hit control+c anytime to enter REPL.  
> |
```

2. Return Something to Me!

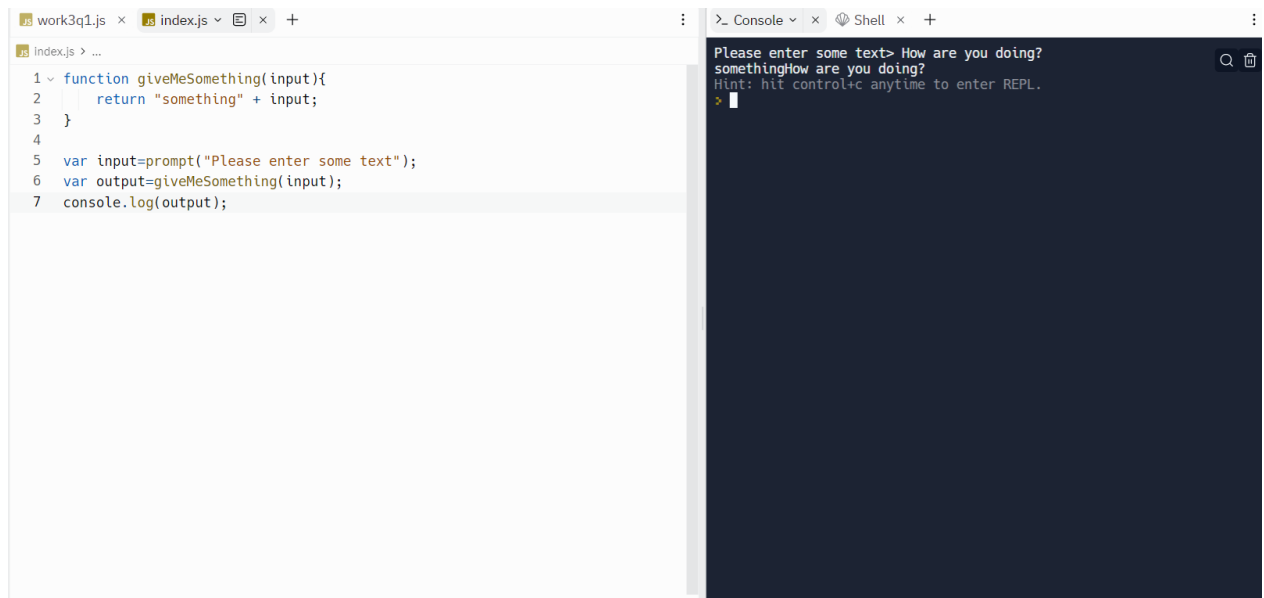
Write a function that returns the string `"something"` joined with a space `" "` and the given argument `a`.

Examples

`giveMeSomething("is better than nothing")` → `"something is better than`

`nothing"` `giveMeSomething("Bob Jane")` → `"something Bob Jane"`

`giveMeSomething("something")` → `"something something"`



The screenshot shows a code editor with a file named `index.js` containing the following JavaScript code:

```
1 function giveMeSomething(input){  
2   return "something" + input;  
3 }  
4  
5 var input=prompt("Please enter some text");  
6 var output=giveMeSomething(input);  
7 console.log(output);
```

To the right of the code editor is a console window. It displays the prompt `Please enter some text>`, followed by the user input `How are you doing?`. The console then shows the output `somethingHow are you doing?` and a hint: `Hint: hit control+c anytime to enter REPL.`

3. Basketball Points

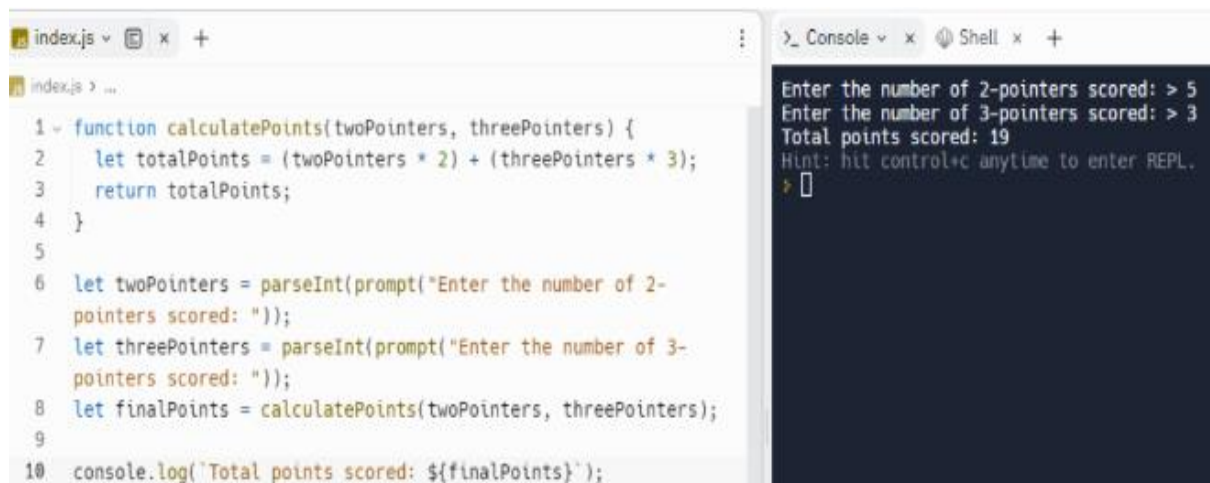
You are counting points for a basketball game, given the amount of 2-pointers scored and 3-pointers scored, find the final points for the team and return that value.

Examples

`points(1, 1) → 5`

`points(7, 5) → 29`

`points(38, 8) → 100`



The image shows a code editor with a file named `index.js` and a terminal window. The code in `index.js` defines a function `calculatePoints` that takes `twoPointers` and `threePointers` as arguments and returns the total points. It then prompts the user for the number of 2-pointers and 3-pointers scored, calculates the total points, and logs the result to the console.

```
1 function calculatePoints(twoPointers, threePointers) {  
2   let totalPoints = (twoPointers * 2) + (threePointers * 3);  
3   return totalPoints;  
4 }  
5  
6 let twoPointers = parseInt(prompt("Enter the number of 2-  
pointers scored: "));  
7 let threePointers = parseInt(prompt("Enter the number of 3-  
pointers scored: "));  
8 let finalPoints = calculatePoints(twoPointers, threePointers);  
9  
10 console.log(`Total points scored: ${finalPoints}`);
```

The terminal window shows the execution of the program. It prompts the user to enter the number of 2-pointers scored (5) and the number of 3-pointers scored (3). The output is "Total points scored: 19". A hint is also displayed: "Hint: hit control+c anytime to enter REPL."

4. Less Than 100?

Given two numbers, return `true` if the sum of both numbers is less than 100.

Otherwise return `false`. **Examples**

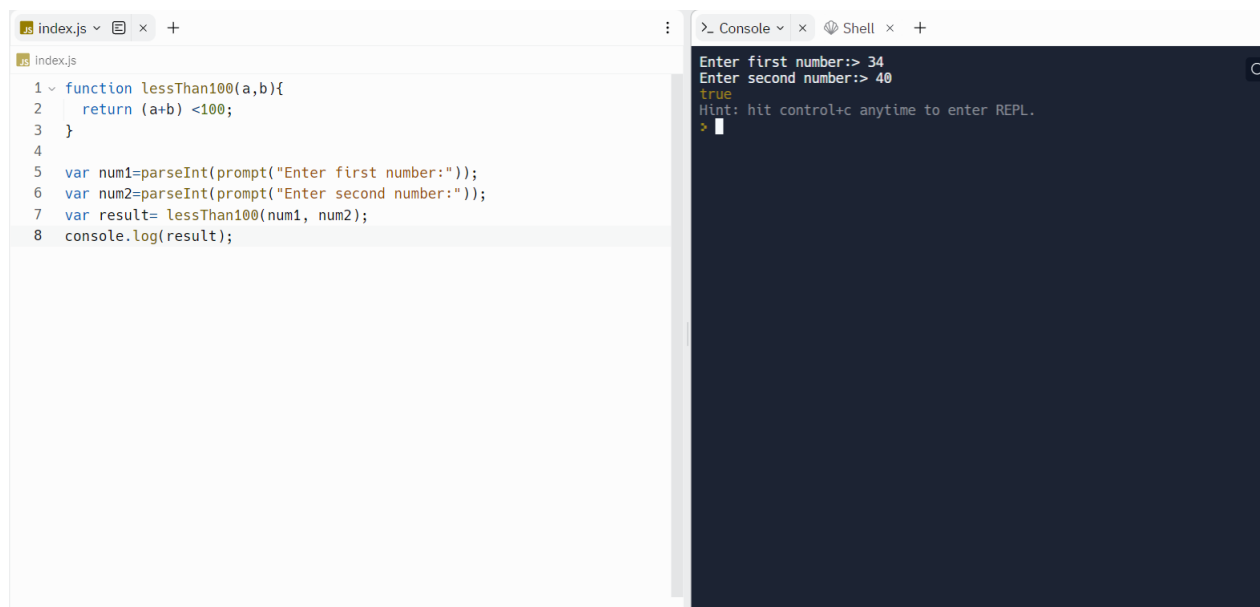
```
lessThan100(22, 15) →true
```

```
// 22 + 15 = 37
```

```
lessThan100(83, 34) →false
```

```
// 83 + 34 = 117
```

```
lessThan100(3, 77) →true
```



The screenshot shows a code editor with a file named `index.js` containing the following JavaScript code:

```
1 function lessThan100(a,b){
2   return (a+b) <100;
3 }
4
5 var num1=parseInt(prompt("Enter first number:"));
6 var num2=parseInt(prompt("Enter second number:"));
7 var result= lessThan100(num1, num2);
8 console.log(result);
```

To the right of the code editor is a terminal window with the following output:

```
>_ Console x Shell x +
Enter first number:> 34
Enter second number:> 40
true
Hint: hit control+c anytime to enter REPL.
> |
```

5.Add up the Numbers from a Single Number

Create a function that takes a number as an argument. Add up all the numbers from 1 to the number you passed to the function. For example, if the input is 4 then your function should return 10 because $1 + 2 + 3 + 4 = 10$.

Examples

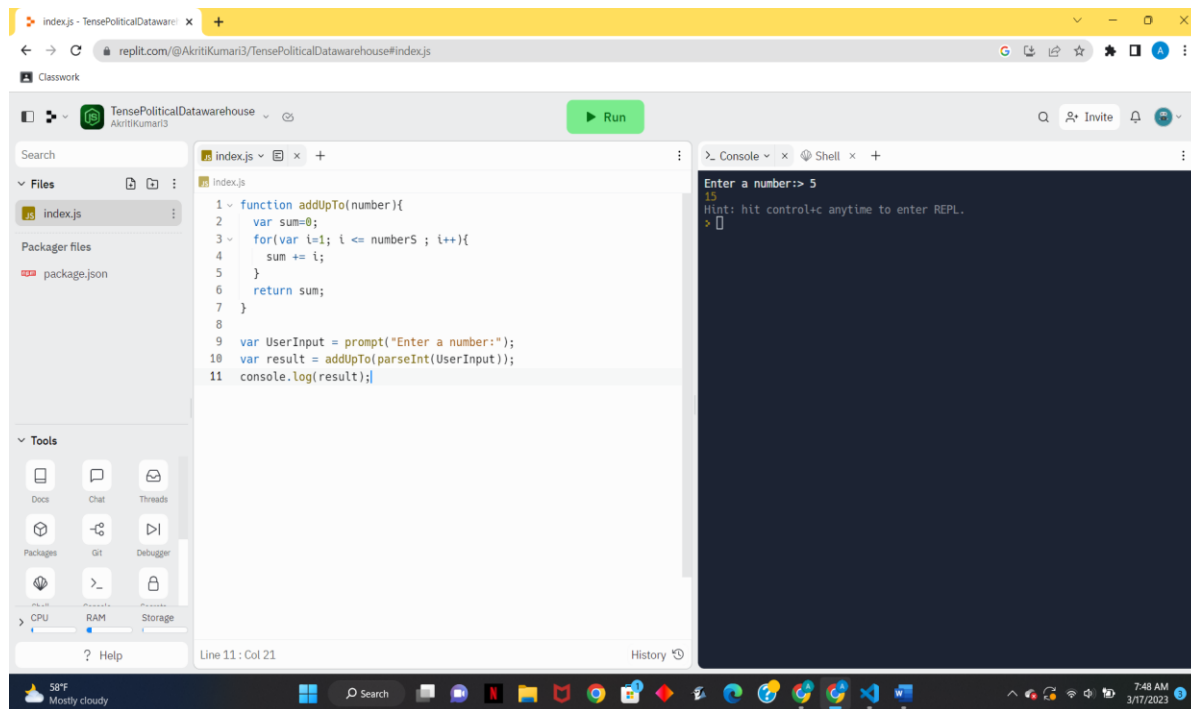
`addUp(4) → 10`

`addUp(13) → 91`

`addUp(600) → 180300`

Notes

Expect any positive number between 1 and 1000.



The screenshot shows a Replit IDE window with a browser tab at `replit.com/@AkritiKumar13/TensePoliticalDatawarehouse#index.js`. The editor displays a JavaScript file named `index.js` with the following code:

```
1 function addUpTo(number){
2   var sum=0;
3   for(var i=1; i <= number; i++){
4     sum += i;
5   }
6   return sum;
7 }
8
9 var UserInput = prompt("Enter a number:");
10 var result = addUpTo(parseInt(UserInput));
11 console.log(result);
```

The console on the right shows the output of the program:

```
Enter a number:> 5
15
Hint: hit control+c anytime to enter REPL.
>
```

The status bar at the bottom indicates the current position is Line 11, Column 21. The system tray shows a temperature of 58°F, mostly cloudy weather, and the date/time as 7:48 AM on 3/17/2023.

6.Oddish vs. Evenish

Create a function that determines whether a number is **Oddish** or **Evenish**. A number is **Oddish** if the sum of all of its digits is odd, and a number is **Evenish** if the sum of all of its digits is even.

If a number is **Oddish**, return "Oddish". Otherwise, return "Evenish".

For example, `oddishOrEvenish(121)` should return "Evenish", since $1 + 2 + 1 =$

4. `oddishOrEvenish(41)` should return "Oddish", since $4 + 1 = 5$. **Examples**

```
oddishOrEvenish(43) → "Oddish"
```

```
// 4 + 3 = 7
```

```
// 7 % 2 = 1
```

```
oddishOrEvenish(373) → "Oddish"
```

```
// 3 + 7 + 3 = 13
```

```
// 13 % 2 = 1
```

```
oddishOrEvenish(4433) → "Evenish"
```

```
// 4 + 4 + 3 + 3 = 14
```

```
// 14 % 2 = 0
```



The screenshot shows a code editor with a file named `index.js`. The code defines a function `oddishOrEvenish(num)` that calculates the sum of digits of `num` using a `while` loop. If the sum is even, it returns "Evenish"; otherwise, it returns "Oddish". The function is then called with `parseInt(prompt("Enter a number:"))`, and the result is logged to the console. The console output shows the prompt "Enter a number: 1345" and the result "Oddish".

```
1 function oddishOrEvenish(num) {
2   var sum = 0;
3   while (num > 0) {
4     sum += num % 10;
5     num = Math.floor(num / 10);
6   }
7   if (sum % 2 == 0) {
8     return "Evenish";
9   } else {
10    return "Oddish";
11  }
12 }
13
14 var userInput = parseInt(prompt("Enter a number:"));
15 console.log(oddishOrEvenish(userInput));
```

Enter a number:> 1345
Oddish
Hint: hit control+c anytime to enter REPL.

7.Any Prime Number in Range

Create a function that returns `true` if there's at least one prime number in the given range

(`n1` to `n2` (inclusive)), `false` otherwise. **Examples**

```
primeInRange(10, 15) →true //
```

Prime numbers in range: 11, 13

```
primeInRange(62, 66) →false
```

```
// No prime numbers in range.
```

```
primeInRange(3, 5) →true
```

```
// Prime numbers in range: 3, 5
```

Notes

- `n2` is always greater than `n1`.
- `n1` and `n2` are always positive.
- 0 and 1 aren't prime numbers.

The screenshot shows a code editor with a file named `index.js` and a terminal window. The code defines two functions: `isPrime(n)` and `primeInRange(n1, n2)`. `isPrime(n)` checks if a number `n` is prime by testing divisibility from 2 to `n`. `primeInRange(n1, n2)` iterates through the range `[n1, n2]` and returns `true` if any prime is found. The main logic prompts the user for two numbers, `n1` and `n2`, and logs a message if a prime is found in the range.

```
1 function isPrime(n) {
2   if (n <= 1) {
3     return false;
4   }
5   for (let i = 2; i * i <= n; i++) {
6     if (n % i === 0) {
7       return false;
8     }
9   }
10  return true;
11 }
12
13 function primeInRange(n1, n2) {
14   for (let i = n1; i <= n2; i++) {
15     if (isPrime(i)) {
16       return true;
17     }
18   }
19   return false;
20 }
21
22 let n1 = parseInt(prompt("Enter first number: "));
23 let n2 = parseInt(prompt("Enter second number: "));
24
25 if (primeInRange(n1, n2)) {
26   console.log("There is at least one prime number in the given
```

The terminal window shows the execution of the program. It prompts for the first number (10) and the second number (20), then outputs: "There is at least one prime number in the given range." A hint at the bottom suggests using `control+c` to enter REPL.

This screenshot shows the same code editor and terminal window, but with additional logic. The `if` statement in the main logic is expanded to include an `else` clause that logs a message when no prime is found in the range.

```
15   if (isPrime(i)) {
16     return true;
17   }
18 }
19 return false;
20 }
21
22 let n1 = parseInt(prompt("Enter first number: "));
23 let n2 = parseInt(prompt("Enter second number: "));
24
25 if (primeInRange(n1, n2)) {
26   console.log("There is at least one prime number in the given
27   range.");
28 } else {
29   console.log("There is no prime number in the given range.");
30 }
```

The terminal window shows the same execution as before, but now it also displays the message: "There is no prime number in the given range." when the range `[10, 20]` is checked.

8. Left Shift by Powers of Two

The left shift operation is similar to multiplication by powers of two.

Sample calculation using the left shift operator (`<<`):

$$10 \ll 3 = 10 * 2^3 = 10 * 8 = 80$$

$$-32 \ll 2 = -32 * 2^2 = -32 * 4 = -128$$

$$5 \ll 2 = 5 * 2^2 = 5 * 4 = 20$$

Write a function that mimics (without the use of `<<`) the left shift operator and returns the result from the two given integers.

Examples

`shiftToLeft(5, 2)` → 20

`shiftToLeft(10, 3)` → 80

`shiftToLeft(-32, 2)` → -128

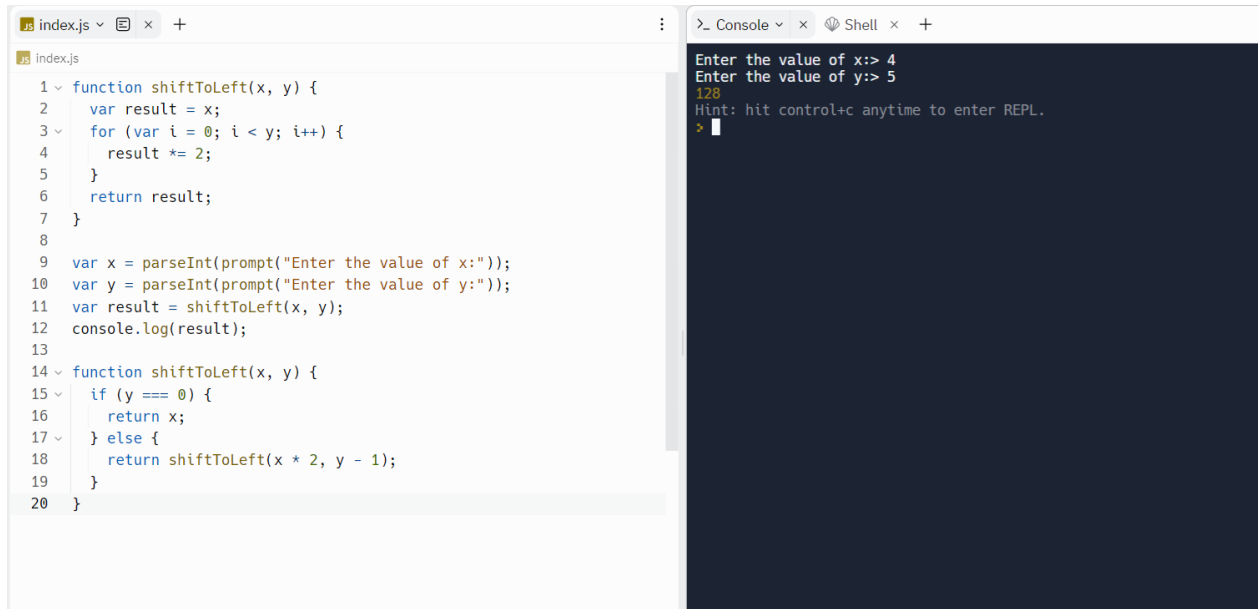
`shiftToLeft(-6, 5)` → -192

`shiftToLeft(12, 4)` → 192

`shiftToLeft(46, 6)` → 2944

Notes

- There will be no negative values for the second parameter `y`.
- This challenge is more like recreating the left shift operation, thus, the use of the operator directly is prohibited.
- Alternatively, you can solve this challenge via recursion.



The image shows a code editor with a file named `index.js` and a terminal window. The code in `index.js` defines a function `shiftToLeft(x, y)` that shifts the bits of `x` to the left by `y` positions. It uses a loop to multiply `result` by 2 for each shift. The main code prompts the user for values of `x` and `y`, calls the function, and logs the result. The terminal shows the user inputting `4` for `x` and `5` for `y`, and the output is `128`.

```
1 function shiftToLeft(x, y) {
2   var result = x;
3   for (var i = 0; i < y; i++) {
4     result *= 2;
5   }
6   return result;
7 }
8
9 var x = parseInt(prompt("Enter the value of x:"));
10 var y = parseInt(prompt("Enter the value of y:"));
11 var result = shiftToLeft(x, y);
12 console.log(result);
13
14 function shiftToLeft(x, y) {
15   if (y === 0) {
16     return x;
17   } else {
18     return shiftToLeft(x * 2, y - 1);
19   }
20 }
```

Enter the value of x:> 4
Enter the value of y:> 5
128
Hint: hit control+c anytime to enter REPL.
>

9.Convert a Number to Base-2

Create a function that returns a base-2 (binary) representation of a base-10 (decimal) string number. To convert is simple: ((2) means base-2 and (10) means base-10) $010101001(2) = 1 + 8 + 32 + 128$.

Going from right to left, the value of the most right bit is 1, now from that every bit to the left will be $\times 2$. The values of an 8 bit binary number are (256, 128, 64, 32, 16, 8, 4, 2, 1).

Examples

`binary(1) → "1"`

```
// 1*1 = 1
```

`binary(5) → "101"`

`// 1*1 + 1*4 = 5`

`binary(10) → "1010"`

`// 1*2 + 1*8 = 10`

Notes

- Numbers will always be below 1024 (not including 1024).
- The `&&` operator could be useful.
- The strings will always go to the length at which the most left bit's value gets bigger than the number in decimal.
- If a binary conversion for 0 is attempted, return "0".



The image shows a code editor with a file named `index.js` and a terminal window. The code in `index.js` defines a `binary` function that converts a decimal number to a binary string. It uses a loop to divide the decimal by 2 and build the binary string from right to left. The terminal shows the function being called with the input 4, resulting in the output "00000100".

```
1 function binary(decimal) {
2   if (decimal == 0) return "0";
3
4   let binary = "";
5
6   for (let value = 256; value >= 1; value /= 2) {
7     if (decimal >= value) {
8       decimal -= value;
9       binary += "1";
10    } else {
11      binary += "0";
12    }
13  }
14
15  return binary;
16 }
17
18 let decimal = prompt("Enter a decimal number");
19 let binaryStr = binary(decimal);
20 console.log(binaryStr);
```

Terminal output:

```
Enter a decimal number> 4
00000100
Hint: hit control+c anytime to enter REPL.
>
```

