**Problem Solving Approach**

**STEP: 1** Understand the Problem Properly

**STEP: 2** Understand the basic solution behind it

**STEP: 3** Analyse input and output with Data Structure

**STEP: 4** Think about which loop, statement can help

**STEP: 5** Do it Step by Step

**Example: //Find even number out of 1 - 10**

**STEP: 1** I know the problem clearly say that I need to find out even numbers from 1 to 10.

**STEP: 2** Let’s, see the basic solution: even numbers means any number which is completely **divisible by 2** i.e. remainder should be zero when divide with 2 & to get remainder we use modulus operator i.e. %

**STEP: 3** Here, input should be an *array of numbers* like [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] & Output 🡪 [2, 4, 6, 8, 10]

**STEP: 4** to find out whether the given no. is even or odd I need to divide it by 2 to all which means I need to run a loop on my array and execute a if condition statement inside in it to check whether the no. is divisible by 2 or not.

**STEP: 5** I will run a loop.

Array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

evenNo = [];

for(let item of Array){

**// now I need to check condition**

If(item % 2 === 0){

evenNo.push(item);  
}

}

**Output:**

[2, 4, 6, 8, 10]

**What is an Algorithm?**

A **process** or **set of steps** to accomplish a certain task.

**Why we need to know this?**Almost **everything** that you do in programming involves some kind of **algorithm**!  
**It’s the foundation for being a successful problem slover and developer.**

**How do you improve?  
1**. **Devise** a plan for solving problems

2. **Master**common problem solving patterns

**Problem Solving Strategies?**

1. Understand the Problem
2. Explore Concrete Examples
3. Break It Down
4. Solve/Simplify
5. Look Back and Refactor

Note: many of these strategies are adapted from [**George Polya**](https://en.wikipedia.org/wiki/George_P%C3%B3lya), whose book How To Solve It is a great resource for anyone who wants to become a better problem solver

**Strategies:   
#1 - Understand the Problem:**

1. Can I restate the problem in my own words?
2. What are the inputs that go into the problem?
3. What are the outputs that should come from the solution to the problem?
4. Can the outputs be determined from the inputs? In other words, do I have enough information to solve the problem? (You may not be able to answer this question until you set about solving the problem. That's okay; it's still worth considering the question at this early stage.)
5. How should I label the important pieces of data that are a part of the problem?

**Example:**

**Write a function which takes two numbers and returns their sum.**

# Solve:

1. Can I restate the problem in my own words?  
   ***‘’implement addition”***
2. What are the inputs that go into the problem?  
   ***-int?***  
   ***-floats?  
   -What about strings for super large numbers?***
3. What are the outputs that should come from the solution to the problem?  
   ***-int? float? string?***
4. Can the outputs be determined from the inputs? In other words, do I have enough information to solve the problem? (You may not be able to answer this question until you set about solving the problem. That's okay; it's still worth considering the question at this early stage.)
5. How should I label the important pieces of data that are a part of the problem?

So this was only the first step, but there were still multiple questions that we can ask ourselves or ask our interviewer about the problem.

**#2 – Explore Concrete Examples:**

Coming up with examples can help you understand the problem better.

Examples also provide sanity checks that your eventual solution works how it should.

**User Stories!** **Unit Tests!**

**Explore Examples….**

1. Start with Simple Examples
2. Progress to More Complex Examples
3. Explore Examples with Empty Inputs
4. Explore Examples with Invalid Inputs

**Example:**

**Write a function which takes in a string and returns counts of each character in the string.**

**Solve:**

It needs to return an object. Let’s say, instead of an array or some other data structure.

1. Start with Simple Examples  
   charCount(“aaaa”); //{a:4}  
   charCount(“hello”); //{h:1, e:1, l:2, o:1}
2. Progress to More Complex Examples  
   charCount(“my phone number is **182934**”);   
   charCount(“**H**ello **h**i”);   
   Consider numbers, spaces, UpperCase, Lower Case, Character Entity about how to store and how differently should it be store.
3. Explore Examples with Empty Inputs

charCount(“”);  
Now what should I return? Should I return an empty object {} at the end or null, or false or undefined or may be an error?

1. Explore Examples with Invalid Inputs  
   What if somebody passes in something that isn’t a string they pass in a number or they pass in an object or they pass in null? There’ all sorts of these edge cases and understanding theses cases in an interview setting, often it’s not that important usually, but it can be really important in the real world to help you to build a more robust solution, something that is more foolproof.

**#3 – Break it down:**

Explicitly write out the steps you need to take.  
It doesn’t have to be line by line, every line you need to write just the basic components of the solution.

This forces you to think about the code you’ll write before you write it, and helps you catch any lingering conceptual issues or misunderstandings before you dive in and have to worry about details (e.g. language syntax) as well.

**Example:**

**Write a function which takes in a string and returns counts of each character in the string.**

**Solve:**

So just to clarify, I decided or my interviewer decided that we only need to care about alphanumeric characters, lowercase as well.

*Here, we’re keeping track of the numbers and we’re keeping track of all letters, but only lowercase them. Neither spaces nor any character entity will included (!).*  
charCount(“Your PIN number is 1234!”)  
/\*{  
1: 1,  
2: 1,  
3: 1,  
4: 1,  
b: 1,  
e: 1,  
 i: 2,  
m: 1,  
n: 2,  
o: 1,  
p: 1,  
r: 2,  
s: 1,  
u: 2,  
y: 1,  
}\*/

function charCount(str){

//do something  
//return an object with keys that are lowercase alphanumeric characters in the string; values should be the counts for those characters

}  
The login involves looping over every character in our string and doing something.

function charCount(str){

//make object to return at end  
//loop over string, for each character….  
 //if the char is a number/letter AND is a key in object, add one to count  
 //if the char is a number/letter AND not in object, add it to object and set value to 1  
 //If character is something else (space, period, !, etc.) don’t do anything  
//return object at end

}

**#4 – Solve / Simplify:**

Solve the Problem and If you can’t, Solve a Simpler Problem!. Now, that doesn’t meant changing the subject entirely.

**Simplify 🡪**

1. Find the core difficulty in what you're trying to do
2. Temporarily ignore that difficulty
3. Write a simplified solution
4. Then incorporate that difficulty back in

**Example:**

**Write a function which takes in a string and returns counts of each character in the string.**

**Solve:**

Firsly, we’re going to ignore char is number or letter, character of lowercase or uppercase & character entity or space.

function charCount(str){

//make object to return at end  
 var result = {};

//loop over string, for each character...

for(let i=0; i<str.length;i++){ **//str.length e.g. “Hello” =5**  
 var char = str[i].toLowerCase(); ***//storing each character to lowercase using their index number.***

**//if the char is a number/letter AND is a key in object, add one to count**  
 if(result[char]>0){ **//result[char] = result[“H”] = undefined at start**  
 result[char]++;  
 }

**//if the char is a number/letter AND not in object, add it to object and set value to 1** else{  
 result[char] = 1; **//result[char] = result[“H”] = 1 e.g. result{h: 1}**  
 };

}   
***//Along with the execution of loop, key & values getting stored into the object i.e. result{}***

**//If character is something else (space, period, !, etc.) don’t do anything  
//return object at end**

return result; //{h: 1, e: 1, l: 2, o: 1, !: 1}  
**//return result object, only after completion of the loop that will return back to the function from where it was calling.**

}

console.log(charCount("Hello"));  
**//since, function returning the result object back to this calling function. So we must have to print that return value i.e. Key & values of return{} Object to the console, either directly calling function into the console or by taking the reference variable calling function.**

**Output:**{h: 1, e: 1, l: 2, o: 1, !: 1} **//result is non-alphanumerical**

**Note:**Here, using result[char], basically we’re giving the value of that particular character (as key). Here we used values of particular keys either for comparison under if condition or for assigning 1 as value under else block for that key particular key.

**#4 – Look Back & Refactor:**

In an interview setting, it’s really important to ask the following questions often out loud.

It’s worth analyzing your code and reflecting on it, looking back on it, rather than simply writing something that works and calling it a day.

**Refactoring Question:**

* Can you check the result?
* Can you derive the result differently?
* Can you understand it at a glance?
* Can you use the result or method for some other problem?
* Can you improve the performance of your solution?
* Can you think of other ways to refactor?
* How have other people solved this problem?

*Solved using forof loop just for simple syntax & used charCodeAt instead of regular Expression.*

function charCount(str){

//make object to return at end  
 var result = {};

//loop over string, for each character...

for(let char of str){ **//for of loop**   
 char = char.toLowerCase(); ***//storing each character to lowercase using their index number.***

**//if the char is a number/letter AND is a key in object, add one to count**  
 if(isAlphaNumeric(char)){  
 **// /[a-z0-9]/.test(char).test() method of expression to test char should be alphanumerical. So, Instead of using regular expression we use to call function of charCodeAt which is more optimized than regular expression**

result[char]= ++result[char] || 1;  
**//result[char] = value of any particular char / key, if undefined i.e. there will be no value then just 1 will assign to it (||1) otherwise if any value is already there (like: 1) for any particular char / key then increment the value of that particular key by 1 (++result[char]).**  
 }

}

return result; //{h: 2, e: 1, l: 2, o: 1, i: 1}

}

console.log(charCount("Hello hi!"));

function isAlphaNumeric(char){

let code = char.charCodeAt(0);

if(!(code > 47 && code < 58) && ***// numeric (0-9)***  
 !(code > 64 && code < 91) && ***// upper alpha (A-Z)***  
 !(code > 96 && code < 123)){ ***// lower alpha (a-z)***

return false;  
 }  
 return true;

}

**Output:**{h: 2, e: 1, l: 2, o: 1, i: 1} **//result is alphanumerical**