Given CODE SNIPPET:

```haskell
isValidParentheses :: String -> Bool
isValidParentheses str = check str 0
  where
    check [] 0 = True
    check [] _ = False
    check (')':xs) n = n > 0 && check xs (n - 1)
    check ('(':xs) n = check xs (n + 1)
    check (_:xs) n = check xs n
main = do
  let result = isValidParentheses "(()"
  print result
```

Given PLAN:
STEP 1: Define a function isValidParentheses which
takes a string as input and produces a boolean as output.
STEP 2: In the main function, let the "result" variable
hold the output of isValidParentheses given the string
"(()".
STEP 3: In the isValidParentheses function, initiate the
conditional checks with parameters str and 0.
STEP 4: Define five checks. Check 1 verifies whether the
parameters are an empty list and the value 0. Check 2 verifies
whether the parameters are an empty list and any value other
than 0. Check 3 verifies whether the first character of the
first parameter is a right parenthesis. Check 4 verifies
whether the first character of the first parameter is a left
parenthesis. Check 5 verifies whether the first character of
the first parameter is any character other than a right or a
left parenthesis. These checks are executed in the order in
which they appear. We stop executing the next check as soon as
we find one which satisfies the correct condition. In the case
of the "(()" input string, given the ordering of the checks,
all checks are executed.
STEP 5: In the main function, print the resulting array to the
screen.

So the CODE COVERAGE for the given code snippet will be:

```
> isValidParentheses :: String -> Bool
> isValidParentheses str = check str 0
>   where
>     check [] 0 = True
>     check [] _ = False
>     check (')':xs) n = n > 0 && check xs (n - 1)
>     check ('(':xs) n = check xs (n + 1)
>     check (_:xs) n = check xs n
> main = do
>   let result = isValidParentheses "(()"
>   print result
```