I have now finished modifying the code and it is in a compile free state.

I didn't expect it to be functional logically so it is perfect opportunity to understand.

TEST CASE 1:

I expected it to find a match in both of these scenarios

```
String s  = "abcdabed";
String p="abd";
```

I have performed a search through all my screen outputs and there is no a single entry referring to this block of code:

```
if (sb.toString().isEmpty() && hasCharFound && !hasIncompleteLettersInStringS)
{
    //We need to simply have lastIndexLocationStringS as the substring since this is last position
    //where it has found a character from String p
    System.out.println("*********************************************************");
    System.out.println(s.substring(startPos,(lastIndexLocationStringS))
    +" is a substring containing characters matching permutation: "+ valuesSet[entry] + "\t\t\tIndex("+counter+")");
    System.out.println("*********************************************************");
```

My best technique is just to simply analyse the applicable startPos and follow logic of my code:

```
String (s) to search in: abcdabed
String (p) template word: abd
*************INITIAL VALUE OF  CYCLES: 0
******************Contents of the backup set
******************Contents of the valuesSet
*************NEW VALUE CYCLES: 19
*************RUNNING TOTAL CYCLES: 19
***PROCESSING SET AT INDEX: 0
**ENDING AT INDEX:***** 6

Checking character: d  against the main String index: 0(a)
SUBSTRING EXAMINED: abc
Checking character: d  against the main String index: 1(b)
SUBSTRING EXAMINED: abc
Checking character: d  against the main String index: 2(c)
SUBSTRING EXAMINED: abc
Checking character: d  against the main String index: 3(d)
SUBSTRING EXAMINED: abc
char found: d    at index: 3
last index location: 3
d has been removed from StringBuilder (String p)= dba
This is current StringBuilder (String p): ba
Checking character: b  against the main String index: 4(a)
SUBSTRING EXAMINED: abc
Checking character: b  against the main String index: 5(b)
SUBSTRING EXAMINED: abc
char found: b    at index: 5
last index location: 5
b has been removed from StringBuilder (String p)= ba
This is current StringBuilder (String p): a
Checking character: a  against the main String index: 6(e)
SUBSTRING EXAMINED: abc
Checking character: a  against the main String index: 7(d)
SUBSTRING EXAMINED: abc
FOLLOWING LETTERS NOT MATCHED: a
StringBuilder being emptied: a
*********************************************************
abcdabed is NOT a substring containing characters matching permutation:: dba                    Index(0)
*********************************************************
```

I can see an extreme lack of clarity on the actual permutation of String p that is being used...

This is actually first time we actually know the permutation.. And we can see it can readily cause confusion with what is defined as part of the challenge

We can see that substring examined should infact be entire substring String s from startPos onwards... This is incorrect

Good news is that we know that this statement is correct. But we can potentially add extra logic in the code.. We know if the number of characters left to analyse in String s is actually longer than String p, we can move to the next iteration (use next permutation available).

I will change my test case as this to verify the insufficient substring. Since it would find 'a' as penultimate character and unable to seek 'b' and 'd'

```
String s  = "abcdabead";
String p="abd";
```

TEST CASE 2:

Implementing changes:

```
String s = "abcdabead";
String p="abd";
```

```
System.out.println("String (s) to search in: " + s);
System.out.println("String (p) template permutation word: " + sb);
```

```
//This is now incorrect. We want to examine everything from startPos onwards
System.out.println("SUBSTRING EXAMINED: " + s.substring(startPos));

            if(sb.toString().length()>(s.length()-startPos))
            {
                System.out.println("Insufficient characters in String s");
                System.out.println("FOLLOWING LETTERS NOT MATCHED: " + sb);
                break;
            }
```

I will now run this test case and see if it reaches insufficient characters in String s

String (s) to search in: abcdabead

String (p) template permutation word: abd //I have taken this example since it matches template word, but could have easily taken adb

SUBSTRING EXAMINED: abc //I am not sure why its showing this as substring since we expected it to be every character from startPos. Even when scrolling down this screen output. I have expressed this in my code.. Since it's a massive screen output, I have remediated code and completed screen output again

```
//for unknown reason, if the end argument is not passed in, it only retreives three characters from the substring
//System.out.println("SUBSTRING EXAMINED: " + s.substring(startPos);

//So I required following remediation
System.out.println("SUBSTRING EXAMINED: " + s.substring(startPos,s.length()));
```

TEST CASE 3:

String (s) to search in: abcdabead

String (p) template permutation word: abd

Checking character: a against the main String index: 0(a)

SUBSTRING EXAMINED: abcdabead

char found: a at index: 0 //This is correct

last index location: 0

a has been removed from StringBuilder (String p)= abd //This is correct

This is current StringBuilder (String p): bd

Checking character: b against the main String index: 1(b)

SUBSTRING EXAMINED: abcdabead

char found: b at index: 1 //This is correct

last index location: 1

b has been removed from StringBuilder (String p)= bd //This is correct

This is current StringBuilder (String p): d

Checking character: d against the main String index: 2(c)

SUBSTRING EXAMINED: abcdabead

Checking character: d against the main String index: 3(d)

SUBSTRING EXAMINED: abcdabead

char found: d at index: 3 //This is correct

last index location: 3

d has been removed from StringBuilder (String p)= d //This is correct

This is current StringBuilder (String p):

****

abc is a substring containing characters matching permutation: abd Index(0) //There is a mistake here, it should mention abcd and not abc
//We know that is infact since last index is exclusion, so need to increase the termination by 1

****

This is minimum window: 9 //more clarity if stated current minimum window

String p (all characters) found within: String s (index: 0 - 3) //This is correct

----------STORING WINDOW RESULT: 3); // //this is incorrect.. We need to instead perform subtraction
(lastIndexLocationStringS-startPos) + 1 due to zero indexing
ROW NUMBER: 1

I have managed to include all my above thought process into the code...
The only aspect not analysed and perhaps it's the most basic....

The biggest structural change was when running the first permutation template word and finding no matches in String s.

Also when there were insufficient characters left in String s to ensure that characters of the template word could be matched...