Solution Explanation: Steps & Test Cases

Solution Explanation:

The given program performs multiple string manipulations based on user inputs. The steps are as follows:

Step 1: Take User Inputs

- The program first collects four inputs from the user:
  1. **Main String** – The base string for operations.
  2. **Substring** – A substring to check for its existence in the main string.
  3. **Character to be replaced** – A character in the main string that needs to be replaced.
  4. **Replacement Character** – The character that will replace the old character.

Step 2: Perform String Operations

- The program performs the following operations using separate functions:
  1. **Check Substring Existence (`CheckSubstringExists`)**
     - Uses `Contains()` method to check if the substring exists in the main string.
  2. **Replace Character (`ReplaceCharacter`)**
     - Uses `Replace()` method to swap the given character with another.
  3. **Swap Case (`SwapCase`)**
     - Iterates over each character in the string.
     - Converts uppercase letters to lowercase and vice versa.
  4. **Remove Whitespaces (`RemoveWhitespace`)**
     - Removes all whitespace characters using `Where()` and `string.Concat()`.
  5. **Count Letter Frequency (`CountLetters`)**
     - Uses `GroupBy()` and `ToDictionary()` to count occurrences of each letter.

Step 3: Display Results

- The program prints:

- Whether the substring exists.
- The string after replacing the character.
- The case-swapped version of the string.
- The string with spaces removed.
- The count of each letter in the main string.

Test Cases

Test Case #	Main String	Substring	Char to Replace	Replacement Char	Expected Output
1	`"Hello World"`	`"World"`	`'o'`	`'x'`	Substring Exists: **Yes** Replaced: `"Hellx Wxrld"` Case Swapped: `"hELLO wORLD"` No Spaces: `"HelloWorld"` Letter Count: `H:1, e:1, l:3, o:2, W:1, r:1, d:1`
2	`"CSharp Programming"`	`"Java"`	`'a'`	`'@'`	Substring Exists: **No** Replaced: `"CSh@rp Progr@mming"` Case Swapped: `"cSHARP pROGRAMMING"` No Spaces: `"CSharpProgramming"` Letter Count: `C:1, S:1, h:1, a:2, r:2, p:2, P:1, o:1, g:2, m:2, i:1, n:1`
3	`"Data Science"`	`"Sci"`	`'e'`	`'3'`	Substring Exists: **Yes** Replaced: `"Data Sci3nc3"` Case Swapped: `"dATA sCIENCE"` No Spaces: `"DataScience"` Letter Count: `D:1, a:2, t:1, S:1, c:2, i:2, e:2, n:1`

4	`"Machine Learning"`	`"Learn "`	`'n'`	`'#'`	Substring Exists: **Yes** Replaced: `"Machine Lear#i#g"` Case Swapped: `"mACHINE lEARNING"` No Spaces: `"MachineLearning"` Letter Count: `M:1, a:2, c:1, h:1, i:2, n:2, e:2, L:1, r:1, g:1`
5	`"Big Data Analytics "`	`"Data"`	`'A'`	`'@'`	Substring Exists: **Yes** Replaced: `"Big Data @nalytics"` Case Swapped: `"bIG dATA aNALYTICS"` No Spaces: `"BigDataAnalytics"` Letter Count: `B:1, i:2, g:1, D:1, a:4, t:1, A:2, n:1, l:1, y:1, c:1, s:1`

Key Observations

- The **substring existence check** correctly identifies whether a substring exists.
- The **replacement operation** changes only the specified character occurrences.
- **Case swapping** accurately toggles upper and lower case.
- **Whitespace removal** removes all spaces without affecting other characters.
- **Letter counting** provides correct frequencies for each letter.

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
    {
      //Input 1: Main String
```

```
        string mainString = GetInput("");

      //Input 2: Sub String
        string substring = GetInput("");

      // Input 3: Character to be replaced
        char charToReplace = GetInput("")[0];

      // Input 4: Character to be replaced with
        char replacementChar = GetInput("")[0];

        bool substringExists = CheckSubstringExists(mainString, substring);
        string replacedString = ReplaceCharacter(mainString, charToReplace,
replacementChar);
        string caseSwapped = SwapCase(mainString);
        string noSpaces = RemoveWhitespace(mainString);
        Dictionary<char, int> letterCount = CountLetters(mainString);

        Console.WriteLine($"Substring Exists: {(substringExists ? "Yes" : "No")}");
        Console.WriteLine($"Replaced: {replacedString}");
        Console.WriteLine($"Case Swapped: {caseSwapped}");
        Console.WriteLine($"No Spaces: {noSpaces}");
        Console.WriteLine($"Letter Count: {string.Join(", ", letterCount.Select(kvp =>
$"{kvp.Key}: {kvp.Value}"))}");
    }

    static string GetInput(string prompt)
    {
        Console.WriteLine(prompt);
        return Console.ReadLine();
    }

    static bool CheckSubstringExists(string main, string sub)
    {
        return main.Contains(sub);
    }

    static string ReplaceCharacter(string input, char oldChar, char newChar)
    {
        return input.Replace(oldChar, newChar);
    }

    static string SwapCase(string input)
    {
        return new string(input.Select(c => char.IsLetter(c) ? (char.IsUpper(c) ?
char.ToLower(c) : char.ToUpper(c)) : c).ToArray());
```

```
    }

    static string RemoveWhitespace(string input)
    {
        return string.Concat(input.Where(c => !char.IsWhiteSpace(c)));
    }

    static Dictionary<char, int> CountLetters(string input)
    {
        return input.GroupBy(c => c).Where(g => char.IsLetter(g.Key)).ToDictionary(g =>
g.Key, g => g.Count());
    }
}
```

Summary of Fixes

Issue	Fix
Empty prompts in `GetInput`	Provide meaningful prompts
Possible `IndexOutOfRangeException`	Validate input length before indexing
Incorrect LINQ in `SwapCase`	Fix syntax in `Select`
Inefficient whitespace removal	Use `string.Concat` with LINQ
Counting all characters in `CountLetters`	Count only letters

There are specific gaps that need to be addressed to ensure a stronger foundation in **C# programming concepts**.

2. Key Issues Identified in Code Submissions

- **Syntax Errors:** Incorrect function declarations, misplaced parentheses, and improper keyword usage.
- **Logical Errors:** Incorrect implementation of string manipulation, improper conditions in loops and if statements.
- **Runtime Issues:** Lack of null checks, leading to potential exceptions.
- **Code Readability:** Inconsistent formatting, making debugging and maintenance difficult.
- **Incorrect Use of C# Features:** Misuse of `Select()`, `ToArray()`, and `ContainsKey()` functions.

3. Improvement Plan & Corrective Actions

To improve the overall coding proficiency and test scores, the following steps will be implemented:

✅ Targeted Training on Core C# Concepts

- Conduct focused sessions on **String Manipulation, Collections, LINQ, and Exception Handling**.
- Reinforce understanding of syntax, logic formulation, and debugging techniques.

✅ Hands-on Practice & Code Reviews

- Assign **structured coding exercises** that reflect common mistakes identified in the test.
- Conduct **peer code reviews** to help participants identify and correct errors.

✅ Guidelines for Future Code Submissions

- Ensure proper syntax and logical structure before submitting.
- Implement **null checks and validation** to avoid runtime errors.
- Follow **consistent naming conventions** for variables and functions.
- Use **comments and proper indentation** for better readability.

✅ Mock Test Retake with Enhanced Evaluation

- Conduct another **mock assessment** after implementing these improvements.
- Include **live debugging sessions** where incorrect answers are analyzed.

4. Next Steps

- A **follow-up training schedule** will be shared.

- Teams are encouraged to **actively participate in coding discussions** to clarify doubts.
- Regular feedback loops will be maintained to track progress.

```
using System;
using System.Collection.Generic;
using System.Linq;
class Program{
  static void Main(){
    //input:Main String
    Console.Write("Enter the main string: ");
    string mainString=Console.ReadLine();
    //input: Substring to Check
    Console.Write("Enter the substring to check: ");
    //input : character to replace
    Console.Write("Enter the character to replace: ");
    char charToReplace= Console.ReadKey().KeyChar;
    Console.WriteLine();
    //input: Replacement character
    Console.Write("Enter the replacement character: ");
    Console.ReadKey().keyChar;
    Console.WriteLine();
    ////1. check if substring exists
    bool substringExists=
    CheckSubstringExists(mainString,substring);
    Console.WriteLine($"Substring Exists:{(substingExits ? "yes":"No")}");
    //2. Replace character
    string replacedstr=ReplaceCharacter(mainString,charToReplace, replacementChar);
    Console.WritwLine($"Replaced: \"{replacedStr}\"");

    //3 swap case
    string caseSwapped=SwapCase(mainString);
    Console.WriteLine($"Case Swaqpped:\"{caseSwapped}\"");
    // remove white spaces
    string noSpaces= RemoveWhitespace(mainString);
    Console.WriteLine($"No Spaces: \"{noSpaces}\"");
    //5 count letter occurrences
    Dictionary<char,int> letterCount=CountLetters(mainString);
    Console.WriteLine("Letter Count:");
    foreach(var kvp in letterCount.OrderBy(k=>k.key))
    {
      Console.WriteLine($"{kvp.key}:{kvp.value}");

    }



  }
  static bool CheckSubstringExists(String main,string sub){
    return main.Contains(sub);
  }
  static string ReplaceCharacter(string input,char oldChar, char newChar){
    return input.Replace(oldChar,newChar);
  }
  static string SwapCase(string input){
    return new string(input.Select(c=>char.IsLetter(c)?(char.IsUpper(c)?
char.ToLower(c):char.ToUpper(c)):c).ToArray());

  }
  static string RemoveWhitespace(string input){
    return new string(input.Where(c=>!char.IsWhiteSpace(c)).ToArray());
  }
  static Dictionary<char,int>letterCount=new Dictionary<char,int>();
  foreach(char c in input.Where(char.IsLetter)){
    if(letterCount.ContainsKey(c))
          letterCount[c]++;
    else
      letterCount[c]=1;

  }
  return letterCount;
}
```

Summary of Fixes

Issue	Fix
`System.Collection.Generic` (incorrect namespace)	Changed to `System.Collections.Generic`

Missing input variable for `substring`	Added `string substring = Console.ReadLine();`
`replacementChar` not assigned	Stored `Console.ReadKey().KeyChar` in `replacementChar`
Misspelled variable `substingExits`	Fixed to `substringExists`
`Console.WritwLine` typo	Changed to `Console.WriteLine`
Incorrectly declared `letterCount`	Fixed method signature and loop
Incorrect use of `.key`	Changed to `.Key`