**Faculty of Computing**

**SE-314: Software Construction**

**Class: BESE 13AB**

# Lab 07: Recursion

**CLO-03:** Design and develop solutions based on Software Construction principles.  
**CLO-04:** Use modern tools such as Eclipse, NetBeans etc. for software construction.

**Date: 28th Oct 2024**

**Time: 10:00 AM** **- 12:50 PM   
 02:30 PM – 04:50 PM**

**Instructor: Dr. Mehvish Rashid  
Lab Engineer: Mr. Aftab Farooq**

**Introduction:**

# Lab 07: Recursion

Students will have hands-on experience on designing, testing, and implementing recursive problems. Given a scenario, you will write the specifications and implement it by dividing into base case and recursive step. You may design helper methods to simplify your implementations. Write unit tests that check for compliance with the specifications.

## Lab Tasks

**Task 1: Recursive File Search**

**Objective:** The objective of this lab task is to create a Java program that recursively searches for a file within a directory and its subdirectories. This exercise will help you practice the principles of software construction and recursion.

**Instructions:+**

1. Create a Java program that takes two command-line arguments: a directory path and a file name to search for.
2. Implement a recursive function to search for the specified file within the given directory and its subdirectories.
3. The program should display a message when it finds the file, including the full path to the file, or a message indicating that the file was not found.
4. Follow good coding practices, including meaningful variable names, comments, and modular code.
5. Implement error handling to handle cases where the specified directory does not exist or other exceptions may occur.
6. Use appropriate data structures and algorithms to efficiently search through the directory tree.
7. Test your program with different directory paths and file names to ensure its correctness and reliability.

Important: Do not forget to write the specifications and unit tests for the code.

**Optional Enhancements:**

1. Allow the program to search for multiple files in a single run.
2. Implement a feature to count the number of times a specific file appears within the directory and its subdirectories.
3. Provide an option to specify whether the search should be case-sensitive or case-insensitive.

**Code:**

**Recursive\_file\_search:**

import os

import sys

def find\_file(directory, target\_file):

    try:

        entries = os.listdir(directory)

    except FileNotFoundError:

        print(f"Error: The directory '{directory}' does not exist.")

        return None

    except PermissionError:

        print(f"Warning: Permission denied for directory '{directory}'.")

        return None

    except Exception as e:

        print(f"An unexpected error occurred: {e}")

        return None

    for entry in entries:

        entry\_path = os.path.join(directory, entry)

        if os.path.isfile(entry\_path) and entry == target\_file:

            return entry\_path

        elif os.path.isdir(entry\_path):

            found\_path = find\_file(entry\_path, target\_file)

            if found\_path:

                return found\_path  # File found in a subdirectory

    return None

def main():

    if len(sys.argv) != 3:

        print("Usage: python search\_file.py <directory\_path> <file\_name>")

        return

    directory\_path = sys.argv[1]

    file\_name = sys.argv[2]

    result = find\_file(directory\_path, file\_name)

    if result:

        print(f"File found: {result}")

    else:

        print(f"The file '{file\_name}' was not found in the directory '{directory\_path}' or its subdirectories.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Test Cases:**

import os

import tempfile

import pytest

from recursive\_file\_search import find\_file

@pytest.fixture

def setup\_test\_directory():

    """

    Creates a temporary directory structure for testing purposes.

    """

    with tempfile.TemporaryDirectory() as tmp\_dir:

        os.mkdir(os.path.join(tmp\_dir, "subdir1"))

        os.mkdir(os.path.join(tmp\_dir, "subdir2"))

        os.mkdir(os.path.join(tmp\_dir, "subdir1", "subsubdir1"))

        with open(os.path.join(tmp\_dir, "testfile.txt"), "w") as f:

            f.write("This is a test file.")

        with open(os.path.join(tmp\_dir, "subdir1", "testfile1.txt"), "w") as f:

            f.write("This is another test file.")

        with open(os.path.join(tmp\_dir, "subdir2", "testfile2.txt"), "w") as f:

            f.write("This is yet another test file.")

        with open(os.path.join(tmp\_dir, "subdir1", "subsubdir1", "targetfile.txt"), "w") as f:

            f.write("This is the target file.")

        yield tmp\_dir

def test\_find\_file\_exists\_in\_root(setup\_test\_directory):

    tmp\_dir = setup\_test\_directory

    result = find\_file(tmp\_dir, "testfile.txt")

    assert result == os.path.join(tmp\_dir, "testfile.txt")

def test\_find\_file\_exists\_in\_subdirectory(setup\_test\_directory):

    tmp\_dir = setup\_test\_directory

    result = find\_file(tmp\_dir, "testfile1.txt")

    assert result == os.path.join(tmp\_dir, "subdir1", "testfile1.txt")

def test\_find\_file\_exists\_in\_nested\_subdirectory(setup\_test\_directory):

    tmp\_dir = setup\_test\_directory

    result = find\_file(tmp\_dir, "targetfile.txt")

    assert result == os.path.join(tmp\_dir, "subdir1", "subsubdir1", "targetfile.txt")

def test\_file\_not\_found(setup\_test\_directory):

    tmp\_dir = setup\_test\_directory

    result = find\_file(tmp\_dir, "nonexistent.txt")

    assert result is None

def test\_directory\_does\_not\_exist():

    result = find\_file("/non/existent/directory", "testfile.txt")

    assert result is None

def test\_permission\_denied(monkeypatch):

    def mock\_os\_listdir(path):

        raise PermissionError("Permission Denied")

    monkeypatch.setattr(os, "listdir", mock\_os\_listdir)

    result = find\_file("/some/protected/directory", "testfile.txt")

    assert result is None

**Output:**

**A black screen with white text

Description automatically generated**

****

**A screenshot of a computer

Description automatically generated**

**Task 2: Recursive String Permutations**

**Objective:** The objective of this lab task is to create a Java program that generates all permutations of a given string using a recursive algorithm. This exercise will help you practice recursion and algorithm design.

**Instructions:**

1. Create a Java program that generates all permutations of a given string using a recursive function.
2. Implement a recursive function **generatePermutations** that takes a string as input and returns a list of all its permutations.
3. Use a recursive approach to generate permutations. You can consider swapping characters in the string to create different permutations.
4. Follow good coding practices, including meaningful variable names, comments, and modular code.
5. Implement error handling to handle cases where the input string is empty or other exceptions may occur.
6. Analyze the time complexity of the recursive algorithm. How does the time complexity compare to an iterative solution for large strings?

**Optional Enhancements:**

1. Provide an option for the user to choose whether to include or exclude duplicate permutations, as some characters in the input string may be identical.
2. Implement a non-recursive algorithm for generating permutations and compare its performance with the recursive solution for large strings.

**Code:**

**String\_permutations.py:**

def generate\_permutations(string):

    if not string:

        print("Error: The input string is empty.")

        return []

    if len(string) == 1:

        return [string]

    permutations = []

    for i in range(len(string)):

        current\_char = string[i]

        remaining\_string = string[:i] + string[i+1:]

        for perm in generate\_permutations(remaining\_string):

            permutations.append(current\_char + perm)

    return permutations

if \_\_name\_\_ == "\_\_main\_\_":

    user\_input = input("Enter a string to generate its permutations: ")

    try:

        result = generate\_permutations(user\_input)

        if result:

            print(f"Permutations of '{user\_input}':")

            for perm in result:

                print(perm)

    except Exception as e:

        print(f"An unexpected error occurred: {e}")

**String permutation test:**

import pytest

from string\_permutations import generate\_permutations  # replace with the actual module name

def test\_empty\_string():

    """

    Test that an empty string returns an empty list.

    """

    assert generate\_permutations("") == []

def test\_single\_character():

    """

    Test that a single character string returns a list with the string itself.

    """

    assert generate\_permutations("a") == ["a"]

def test\_two\_characters():

    """

    Test that a two-character string returns two permutations.

    """

    result = generate\_permutations("ab")

    assert sorted(result) == sorted(["ab", "ba"])

def test\_three\_characters():

    """

    Test that a three-character string returns six permutations.

    """

    result = generate\_permutations("abc")

    assert sorted(result) == sorted(["abc", "acb", "bac", "bca", "cab", "cba"])

def test\_large\_input():

    """

    Test a slightly larger string to ensure the function does not crash.

    """

    result = generate\_permutations("abcd")

    assert len(result) == 24  # 4! = 24

def test\_non\_alphabetic\_characters():

    """

    Test that the function works with non-alphabetic characters.

    """

    result = generate\_permutations("1a!")

    assert sorted(result) == sorted(["1a!", "1!a", "a1!", "a!1", "!1a", "!a1"])

@pytest.mark.parametrize("input\_str, expected\_length", [

    ("abc", 6),  # 3! permutations

    ("abcd", 24),  # 4! permutations

    ("abcde", 120)  # 5! permutations

])

def test\_permutations\_length(input\_str, expected\_length):

    """

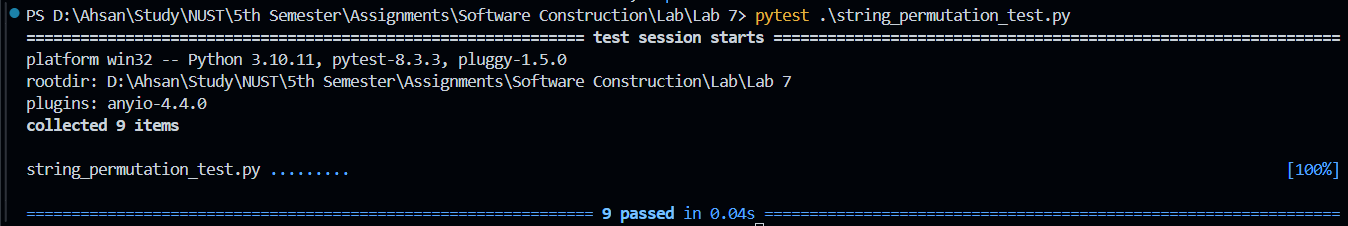
    Parametrized test to check if the number of permutations matches n!.

    """

    result = generate\_permutations(input\_str)

    assert len(result) == expected\_length

**Ouput:**

****

**A screen shot of a computer

Description automatically generated**

### Deliverables:

Compile a single word document by filling in the solution part and submit this Word file on LMS.

In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to [aftab.farooq@seecs.edu.pk.](mailto:aftab.farooq@seecs.edu.pk.)