DATA 3402-Python II

Homework 2- Recursion Functions

Due: 09/10/2025 23:59PM

Assignment Submission Guidelines

Please follow the guidelines below for submitting your assignment:

1. **Submission Deadline:**
   - All assignments must be submitted **no later than 09/10/2025 11:59 PM**.
   - Late submissions will not be accepted unless prior arrangements have been made by the instructor.
2. **Submission Platform:**
   - Submit your assignment through **Canvas**. Ensure that you upload the files to the correct assignment link.
3. **Required Files:**
   - **Jupyter Notebook file (.ipynb):** Submit the Jupyter Notebook file you used to complete the assignment. The file should contain your well-commented code.
   - **PDF Version (.pdf file):** Additionally, submit a PDF version of your Python code. This can be a printout or export of your script, showing all the code with any necessary explanations or output results included.
4. **File Naming Convention:**
   - Please name your files as follows: `Lastname_Firstname_AssignmentName`
   - Example: `Alex_John_Homework2.ipynb` and `Alex_John_Homework2.pdf`
5. **Technical Issues:**
   - If you encounter any technical issues with Canvas or your submission, please contact the TAs immediately **before the deadline** to avoid penalties.

Problem 1

Factorial Calculation – Write a recursive function to compute the factorial of a given number `n`.

Task: Implement the recursive function below.

```python
# Write your recursive solution here
def factorial(n):
    # n! = n*(n-1).
    if n==1:
        return n

    # Where recursion starts.
    else:
        return n*factorial(n-1)
factorial(4)
```

```
24
```

Problem 2

Sum of Digits – Write a recursive function that takes an integer `n` and returns the sum of its digits.

Task: Implement the recursive function below.

```
# Write your recursive solution here
def sum_digits(n):
    if n<=10:
        return n
    else:
        return n % 10 + sum_digits(n // 10)
sum_digits(123)
```

```
6
```

Problem 3

Reverse a String – Write a recursive function to reverse a string without using slicing.

Task: Implement the recursive function below.

```
# Write your recursive solution here
def str_rec(string):
    if isinstance(string, list):
        return string
    else:
        return str_rec(list(reversed(string)))
str_rec('BOAT')
```

```
['T', 'A', 'O', 'B']
```

Problem 4

Power Function – Write a recursive function that computes `x^n` (x to the power of n).

Task: Implement the recursive function below.

```
# Write your recursive solution here
def power(x, n):
    if not isinstance(x, int) and not isinstance(n, int):
        return TypeError("PLEASE ENTER A NUMBER")
    else:
        return x**n
power(2,3)
```

```
8
```

Problem 5

Binary Search – Implement recursive binary search on a sorted list. Return the index if found, otherwise `-1`.

Task: Implement the recursive function below.

```
# pseudo
# Try to find a index position.
# return that index position.
# stop if index position is found

# Write your recursive solution here
sorted_list = [3, 4, 5, 6, 7, 8, 9, 10]
def binary_search(search_q, this_list):
    if search_q not in this_list:
        return -1
    else:
        return [print(f"index location: {n}") for n in range(len(this_list)) if this_list[n]

binary_search(3, sorted_list)
```

```
index location: 0
```

```
[None]
```

Problem 6

Generate All Subsets (Power Set) – Write a recursive function that returns all subsets of a list. Example: [1,2] → [[ ], [1], [2], [1,2]]

Task: Implement the recursive function below.

```
# Write your recursive solution here
## returns the empty position, the 0th position, the nth postion, then an element containin
sub_list = [1,2,3,4]
def subsets(sub_list, empty=[], el_str=[]):
    if sub_list==0:
        return False
    else:
        for el in range(len(sub_list)+1):
            if el==0:
                print("0th: ", el)
                empty.append([])
            else:
                el_str.append(el)
                empty.append([el])

            if el>len(sub_list)-1:
                print("nth: ", el)
                empty.append(el_str)
    return empty

subsets(sub_list)
```

```
0th:  0
nth:  4
```

```
[[], [1], [2], [3], [4], [1, 2, 3, 4]]
```

Problem 7

Sum of List – Compute the sum of all elements in a list recursively.

Task: Implement the recursive function below.

```
# Write your recursive solution here
# Basically, make a summation function for all the elements in number list.
numbers = [1,2,3,3]
def summation(list_to_sum):
    if list_to_sum==None:
        return False
    else:
        return sum(list_to_sum)
summation(numbers)
```

```
9
```

Problem 8

Find Minimum in List – Find the minimum element in a list recursively (without using `min()`).

Task: Implement the recursive function below.

```python
# Write your recursive solution here
numbers = [5,3,2,9,1,2,3,6]
def min_list(find_me):
    if len(find_me)==1:
        return find_me[0]
    else:
        n = min_list(find_me[1:])
        return find_me[0] if find_me[0] < n else n
min_list(numbers)
#[find_me[n] for n in range(len(find_me)) if find_me[n] > n]
```

```
1
```

Problem 9

Decimal to Binary Conversion – Convert a decimal number into binary using recursion.

Task: Implement the recursive function below.

```python
# Write your recursive solution here
def convert(decimal):
    if not isinstance(decimal, float):
        raise TypeError("PLEASE ENTER A DECIMAL NUMBER")
    if isinstance(decimal, int):
        raise TypeError("PLEASE ENTER A DECIMAL NUMBER")
    else:
        return int(decimal)
convert(1.0)
```

```
1
```

Problem 10

String Length – Calculate the length of a string recursively (without using `len()`).

Task: Implement the recursive function below.

```python
# Write your recursive solution here
def str_len(string):
    if string == "":
        return -1
    else:
        return 1 + str_len(string[:-1])
str_len("FRUIT")
```

```
4
```

Problem 11

Nested List Sum – Sum all integers inside a nested list recursively. Example: [1, [2, [3, 4]], 5] → 15.

Task: Implement the recursive function below.

```
# Write your recursive solution here
nested_list = [1, [2, [3, 4]], 5]
def nested_sum(L):
    summ = 0
    for element in L:
        if isinstance(element, list):
            summ += nested_sum(element)
        else:
            summ += element
    return summ
nested_sum(nested_list)
```

```
15
```

Problem 12

Permutations of String – Print all permutations of a string using recursion. Example: "abc" → ['abc', 'acb', 'bac', 'bca', 'cab', 'cba'].

Task: Implement the recursive function below.

```
# Write your recursive solution here
def permutation(string):
    # Base case.
    if len(string) == 1:
        return [string]

    perms = list()

    # Loop that holds the recursion.
    for i in range(len(string)):
        # We slice the string at the ith element, then create remaining as the new string.
        # So remaining = (slice - rest of the word).
        remaining = string[:i] + string[i+1:]

        # letter is the placeholder for the recursive loop.
        # every ith letter gets seperated from the string
        letter = string[i]

        # RECURSION STARTS HERE.
        for perm in permutation(remaining):
            # remaining gets used as the 'string' arguement.
            # creates the new word.
            perms.append(letter + perm)
    return perms

permutation('word')
```

```
['word',
 'wodr',
 'wrod',
 'wrdo',
 'wdor',
 'wdro',
 'owrd',
 'owdr',
 'orwd',
 'ordw',
 'odwr',
 'odrw',
 'rwod',
 'rwdo',
 'rowd',
 'rodw',
 'rdwo',
 'rdow',
 'dwor',
 'dwro',
 'dowr',
 'dorw',
 'drwo',
 'drow']
```