MDS5102 Python Programming

Assignment 1

Due Date: 17 October 2021

Assignment Description:

This assignment will worth **10%** of the final grade.

You should write your code for each question in a .py file (please name it using the question name, e.g. q1.py). Please pack all your .py files into a single .zip file, name it using your student ID (e.g. if your student ID is 123456789 and to submit Assignment 1, then the file should be named as 123456789_Assignment1.zip or 123456789_A1.zip), and then **submit the .zip** file via BlackBoard.

Please also write a report file, which provide the details of your codes. (Note that the report should be submitted as PDF). The report should also be included in the .zip file as well.
Please note that, the teaching assistant may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself. Plagiarism will not be tolerated. We may check your code using Blackboard.

This assignment is due on **23:59PM, 17 Oct (Sunday)**. For each day of late submission, you will lose 10% of your mark in corresponding assignment. If you submit more than three days later than the deadline, you will receive zero in this assignment.

Question 1 **Emirp** (15% of this assignment)

An emirp (prime spelled backward) is a nonpalindromic prime number whose reversal is also a prime. For example, both 17 and 71 are prime numbers, so 17 and 71 are emirps. Write a program that requires user to input an integer N, then displays the first N emirps. Display 10 numbers per line and align the numbers properly, as follows:

```
 13   17   31   37   71   73   79   97  107  113
149  157  167  179  199  311  337  347  359  389
...
```

You can assume N ranges from 0 to 500.

Question 2 **Integral** (15% of this assignment)

Given a function f(x), and a real interval [a, b], the numerical integration of f(x) over interval [a, b] can be calculated as:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \frac{b-a}{n} f\left(a + \frac{b-a}{n} \times \left(i - \frac{1}{2}\right)\right)$$

In equation above, n represents the number of sub-intervals into which the interval [a, b] will be divided; and it controls the accuracy of numerical integration.

Write a program to allow the user to specify a **trigonometric function f** (f can only be sin, cos or tan), and input the **interval end points a, b** and **number of sub-intervals n**. Your program should then calculate the numerical integration of f over [a, b] using equation above, and output the result.
Your program should be robust enough to handle possible improper inputs (e.g. the user inputs a floating point number as n; the user inputs a wrong function name).

Note: Python has built-in trigonometric functions. To call them, use the following statement in your program to import them from the math package:

```
>>> from math import sin
>>> from math import cos
>>> from math import tan
```

You can then invoke the trigonometric functions like the following examples:

```
>>> sin(1)
0.8414709848078965
>>> cos(3.1415)
-0.9999999957076562
>>> tan(0)
0.0
```

For more details about the **math** package, please visit the following link:
https://docs.python.org/3/library/math.html#math.sin

Question 3 **Locker puzzle** (20% of this assignment)
A school has 100 lockers and 100 students. All lockers are closed on the first day of school. As the students enter, the first student S1 opens every locker. Then the second student, S2, begins with the second locker, denoted as L2, and closes every other locker after L2 (which means 1,3,5,7… are opened and 2,4,6,8… are closed). Student S3 begins with the third locker and switches every third locker (closes it if it was open, and opens it if it was closed). Student S4 begins with locker L4 and changes every fourth locker. Student S5 starts with L5 and changes every fifth locker, and so on, until student S100 changes L100.
After all the students have passed through the building and changed the lockers, which lockers are open? Write a program to find your answer, and print all indexes of lockers that are opened. Locker index starts from 1.
(Hint: Use a list of 100 Boolean elements, each of which indicates whether a locker is open (True) or closed (False). Initially, all lockers are closed.)

Question 4 **Binary Tree** (25% of this assignment)
Given a list of integers, build a class of binary tree (each node has at least two child nodes). For each node in the tree, value of its left child is always smaller than its own and value of its right child node is larger than its own.

You are supposed to define classes of Node and BinaryTree.
Node should contain properties of value, left and right children.
BinaryTree must have functions as following,

__init__: initialize the class;

insert: take integer as input. Starting from root node, place input to left side of existing node if input smaller than node value, and place it to right if larger than node value until an empty node found;

search: take an integer as input, return node whose value equals to input, return None if no matched case;

delete: take no input, delete all values in binary tree.

printTree: print the numbers in tree in ascending order (Start with "Values in Tree:\n"; all numbers in one line). Print a "Nothing in Tree" if tree is empty.

Any other helper functions are allowed.

Example code skeleton is shown below.

```
class Node:
    def __init__(self, val):


class Tree:
    def __init__(self):
        """
        initialize root node
        """
        self.root = None


    def getRoot(self):
        """
        return root Node
        """
        return self.root


    def insert(self, val):
        """
        insert value;
        place it to left if larger than current node,
        right if smaller than current value;
        return nothing
        """


    def search(self, val):
        """
        search node by input value;
        return node if value == input, return None otherwise
        """


    def delete(self, val):
        """
        delete node according to input value;
        delete node, whose value equals to input, and its children
        """


    def printTree(self):
        """
        print the values of nodes in tree in ascending order
```

Some example behaviors of defined Binary tree:

```
tree = Tree()
tree.insert(8)
tree.insert(10)
tree.insert(1)
tree.insert(6)
tree.insert(15)
tree.printTree()

tree.delete()
tree.printTree()

content = [1, 5, 99, 15, 100, 35, 23, 20, 16, 13]
for num in content:
    tree.insert(num)
tree.printTree()
```

```
Values of nodes in tree:
1 6 8 10 15

Nothing in tree!

Values of nodes in tree:
1 5 13 15 16 20 23 35 99 100
```

Question 5 **Permutation** (25% of this assignment)
Given a list of integers with no duplicated elements named *numbers*, sorted in ascending order. Print a list of all possible permutations (each element appears only once). You can assume the number of elements in list ranges from 1 to 100.
Define a function named **permute()**, whose input is a list of integers.

Example:

Input: numbers = [3, 5, 9]
permute(numbers)
Output: [[3, 5, 9], [3, 9, 5], [5, 3, 9], [5, 9, 3], [9, 3, 5], [9, 5, 3]]

Input: numbers = [0, 1]
permute(numbers)
Output: [[0, 1], [1, 0]]

Input: numbers = [6]
permute(numbers)
Output: [[6]]

(Hint: Similar with Question 4. construct a tree, then every path from root to leaf is a permutation. Remove paths containing duplicated elements)