



## **Homework #7**

**01286121 Computer Programming  
Software Engineering Program,  
Department of Computer Engineering,  
School of Engineering, KMITL**

By

66011149 Phatthadon Sornplang

1. Define a Clock class in Python, whose properties are hour, minute, second; and it provides methods to set time, get time, tick (increment the current time by 1 second), and display time in am/pm format.

```
class Clock:
```

```
    def __init__(self, hour, minute, second):
```

```
        self.hour = hour
```

```
        self.minute = minute
```

```
        self.second = second
```

```
    def set_time(self, h, m, s):
```

```
        if 0 <= h < 24 and 0 <= m < 60 and 0 <= s < 60:
```

```
            self.hour = h
```

```
            self.minute = m
```

```
            self.second = s
```

```
        else:
```

```
            return "Invalid input"
```

```
    def get_time(self):
```

```
        return self.hour, self.minute, self.second
```

```
    def tick(self):
```

```
        self.second += 1
```

```
        if self.second >= 60:
```

```
            self.second = 0
```

```
            self.minute += 1
```

```
            if self.minute >= 60:
```

```
                self.minute = 0
```

```
                self.hour += 1
```

```
                if self.hour >= 24:
```

```
                    self.hour = 0
```

```
    def display_time(self):
```

```
        am_pm = "AM"
```

```
        hour = self.hour
```

```
        if hour >= 12:
```

```
            am_pm = "PM"
```

```
            if hour > 12:
```

```
                hour -= 12
```

```
        if hour == 0:
```

```
            hour = 12
```

```
        return f"{hour:02d}:{self.minute:02d}:{self.second:02d} {am_pm}"
```

```
time = Clock(15,55,30)
```

```
# print(time.display_time())
```

```
for i in range(86400):
```

```
    time.tick()
```

```
    print(time.display_time())
```

```
03:54:49 PM
03:54:50 PM
03:54:51 PM
03:54:52 PM
03:54:53 PM
03:54:54 PM
03:54:55 PM
03:54:56 PM
03:54:57 PM
03:54:58 PM
03:54:59 PM
03:55:00 PM
03:55:01 PM
03:55:02 PM
03:55:03 PM
```

2. A single-variable polynomial can be represented in python as a tuple of coefficients. For example, the polynomial  $14 + 7x - 5x^2 + 18x^3$  can be represented as (14, 7, -5, 0, 0, 18).

```
def calendar_of_2023(x):
```

```
class Poly:
```

```
    def __init__(self, coefficients):
        self.coefficients = coefficients
```

```
    def add(self, other):
        max_length = max(len(self.coefficients), len(other.coefficients))
        coeffs1 = self.coefficients + (0,) * (max_length - len(self.coefficients))
        coeffs2 = other.coefficients + (0,) * (max_length - len(other.coefficients))
        result_coefs = tuple(a + b for a, b in zip(coeffs1, coeffs2))
        return Poly(result_coefs)
```

```
    def scalar_multiply(self, scalar):
        result_coefs = tuple(coef * scalar for coef in self.coefficients)
        return Poly(result_coefs)
```

```
    def multiply(self, poly1, poly2):
        max_length = len(poly1) + len(poly2) - 1
        result_coefs = [0] * max_length
        for i in range(len(poly1)):
            for j in range(len(poly2)):
                result_coefs[i + j] += poly1[i] * poly2[j]
        return tuple(result_coefs)
```

```
    def power(self, exponent):
        if exponent < 0:
            raise ValueError("Invalid input")

        result_coefs = self.coefficients
        for _ in range(exponent - 1):
            result_coefs = self.multiply(result_coefs, self.coefficients)
        return Poly(result_coefs)
```

```
    def diff(self):
        result_coefs = [0] * max(0, len(self.coefficients) - 1)
        for i in range(1, len(self.coefficients)):
```

```

        result_coefs[i - 1] = i * self.coefficients[i]
    return Poly(result_coefs)

def integral(self):
    result_coefs = [0]
    for i in range(len(self.coefficients)):
        result_coefs.append(self.coefficients[i] / (i + 1))
    return Poly(result_coefs)

def eval(self, x):
    result = 0
    for i in range(len(self.coefficients)):
        result += self.coefficients[i] * (x ** i)
    print(f"{result}")

def print(self):
    terms = []
    for i, coef in enumerate(self.coefficients):
        if coef != 0:
            sign = ""
            if coef < 0:
                sign += "- "
            elif i > 0:
                sign += "+ "
            if abs(coef) != 1 or i == 0:
                sign += str(abs(coef))
            if i > 0:
                sign += "x"
            if i > 1:
                sign += f"^{i}"
            terms.append(sign)

    print(" ".join(terms).rstrip("+"))

p = Poly((1, 0, -2))
p2 = Poly((14, 7, -5, 0, 0, 18))
p.print()
q = p.power(2)
q.print()
p.eval(3)
r = p.add(q)
r.print()
r.diff().print()

```

```

phatt@Macbook_Pro MINGW64 ~/OneDrive/Desktop/Code Files/Python/Computer Programming (Python)/7/HW
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/OneDrive/Desktop/Code Files/Python/Computer Programming (Python)/7/HW/2.py"
1 - 2x^2
1 - 4x^2 + 4x^4
-17
2 - 6x^2 + 4x^4
- 12x + 16x^3

```

3. Design a class named LinearEquation for a 2 X 2 system of linear equations:

```
class LinearEquation:
```

```
    def __init__(self,a,b,c,d,e,f):
```

```
        self.__a = a
```

```
        self.__b = b
```

```
        self.__c = c
```

```
        self.__d = d
```

```
        self.__e = e
```

```
        self.__f = f
```

```
    def geta (self):
```

```
        return self.__a
```

```
    def getb (self):
```

```
        return self.__b
```

```
    def getc (self):
```

```
        return self.__c
```

```
    def getd (self):
```

```
        return self.__d
```

```
    def gete (self):
```

```
        return self.__e
```

```
    def getf (self):
```

```
        return self.__f
```

```
    def isSolvable (self):
```

```
        if (self.__a * self.__d) - (self.__b * self.__c) != 0:
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    def getX(self):
```

```
        if self.isSolvable() == True:
```

```
            return ((self.__e * self.__d) - (self.__b * self.__f)) / ((self.__a * self.__d) - (self.__b * self.__c))
```

```
    def getY(self):
```

```
        if self.isSolvable() == True:
```

```
            return ((self.__a * self.__f) - (self.__e * self.__c)) / ((self.__a * self.__d) - (self.__b * self.__c))
```

```
le = LinearEquation(1,2,3,4,5,6)
```

```
print(le.getX())
```

```
phatt@macbook_Pro MINGW64 ~/OneDrive/Desktop/Code Files/Python/Computer Programming (Python)/HW
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/OneDrive/Desktop/Code
puter Programming (Python)/7/HW/3.py"
-4.0
```