

**Homework #7**

**01286121 Computer Programming**

**Software Engineering Program,**

**Department of Computer Engineering,**

**School of Engineering, KMITL**

By

68011278 Ananda Stallard

1.

Code:

class Clock(object):

def \_\_init\_\_(self, hour, minute, second):

self.hour = hour

self.minute = minute

self.second = second

def set\_time(self, hour = 0, minute = 0, second = 0):

if 24 > hour >= 0 and 60 > minute >= 0 and 60 > second >= 0:

self.hour = hour

self.minute = minute

self.second = second

else:

return print("Please input a valid time")

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(self):

if self.hour == 0:

hour = 12

period = "AM"

elif self.hour < 12:

hour = self.hour

period = "AM"

elif self.hour == 12:

hour = 12

period = "PM"

else:

hour = self.hour - 12

period = "PM"

print(f"{hour:02}:{self.minute:02}:{self.second:02} {period}")

#Testing

time = Clock(12, 0, 0)

time.get\_time()

time.display()

time.tick()

time.display()

time.set\_time(20, 15, 20)

time.get\_time()

time.display()

time.tick()

time.display()

time.set\_time(20, 59, 59)

time.get\_time()

time.display()

time.tick()

time.display()

time.set\_time(26, 68, 18)

Result:

A screenshot of a computer

AI-generated content may be incorrect.

2.

Code:

class Poly(object):

def \_\_init\_\_(self, co):

self.co = list(co)

def add(self, p):

max\_len = max(len(self.co), len(p.co))

new = [0] \* max\_len

for i in range(len(self.co)):

new[i] += self.co[i]

for i in range(len(p.co)):

new[i] += p.co[i]

return Poly(tuple(new))

def scalar\_multiply(self, n):

return Poly(tuple(c \* n for c in self.co))

def multiply(self, p):

new = [0] \* (len(self.co) + len(p.co) - 1)

for i1, x1 in enumerate(self.co):

for i2, x2 in enumerate(p.co):

new[i1 + i2] += x1 \* x2 #i1 + i2 is for the x^ and x1\* x2 is for the num

return Poly(tuple(new))

def power(self, n):

result = Poly((1,))

if n >= 0:

for i in range(n):

result = result.multiply(self)

return result

def diff(self):

if len(self.co) == 1:

return Poly((0,))

new = [i \* c for i, c in enumerate(self.co)][1:]

return Poly(tuple(new))

def integrate(self):

new = [0] \* (len(self.co) + 1)

for i, c in enumerate(self.co):

new[i + 1] = c / (i + 1)

return Poly(tuple(new))

def eval(self, n):

return sum(c \* (n \*\* p) for p, c in enumerate(self.co))

def print(self):

terms = []

for p, c in enumerate(self.co):

if c == 0:

continue

if p == 0:

terms.append(str(c))

elif p == 1:

terms.append(f"{'+ ' if c > 0 else ''}{c}x")

else:

terms.append(f"{'+ ' if c > 0 else ''}{c}x^{p}")

string = " ".join(terms)

string = string.replace("+ -", "- ")

print(string if string else "0")

#Testing

p = Poly((1, 0, -2))

p.print()

q = p.power(2)

q.print()

print(p.eval(3))

r = p.add(q)

r.print()

r.diff().print()

Result:

A black background with white numbers

AI-generated content may be incorrect.

3.

Code:

class LinearEquation(object):

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 get\_a(self): return self.\_\_a

def get\_b(self): return self.\_\_b

def get\_c(self): return self.\_\_c

def get\_d(self): return self.\_\_d

def get\_e(self): return self.\_\_e

def get\_f(self): return self.\_\_f

def isSolvable(self):

if (self.\_\_a \* self.\_\_d) - (self.\_\_b \* self.\_\_c) == 0:

return False

else:

return True

def getX(self):

if self.isSolvable() == False:

return print("This equation is not solvable.")

else:

return ((self.\_\_e \* self.\_\_d) - (self.\_\_b \* self.\_\_f)) / ((self.\_\_a \* self.\_\_d) - (self.\_\_b \* self.\_\_c))

def getY(self):

if self.isSolvable() == False:

return print("This equation is not solvable.")

else:

return ((self.\_\_a \* self.\_\_f) - (self.\_\_e \* self.\_\_c)) / ((self.\_\_a \* self.\_\_d) - (self.\_\_b \* self.\_\_c))

equation = LinearEquation(2, 3, 1, 2, 13, 8)

print("x =", equation.getX())

print("y =", equation.getY())

Result:

A black background with white text

AI-generated content may be incorrect.