



Homework # 7

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

By

65011693 Soe Moe Htet

No1

```
class Clock:
    def __init__(self, hour, minute, second):
        self.hour = hour
        self.minute = minute
        self.second = second

    def set_hour(self, hour):
        self.hour = hour

    def set_minute(self, minute):
        self.minute = minute

    def set_seconds(self, second):
        self.second = second

    def set_time(self, hour, minute, second):
        self.set_hour(hour)
        self.set_minute(minute)
        self.set_seconds(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 get_time(self):
        if(self.hour > 0 and self.hour <= 12):

            state = "AM"
        elif(self.hour > 12 and self.hour < 24):
            self.hour -= 12
            state = "PM"

        elif(self.hour == 12):
            self.hour = 0
            state = "PM"

        elif(self.hour == 24):
            self.hour = 0
            state = "AM"

        print(f"{self.hour:02d}:{self.minute:02d}:{self.second:02d} {state}")
```

```

def main():
    clock = Clock(12, 50, 12)
    clock.set_time(9,30,24)
    clock.set_hour(13)
    clock.set_seconds(30)
    clock.get_time()

main()

```

Sample output:

The screenshot shows the Visual Studio Code editor with a file named `No1.py` open. The code in the file is as follows:

```

36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
def get_time(self):
    if(self.hour > 0 and self.hour <= 12):
        state = "AM"
    elif(self.hour > 12 and self.hour < 24):
        self.hour -= 12
        state = "PM"
    elif(self.hour == 12):
        self.hour = 0
        state = "PM"
    elif(self.hour == 24):
        self.hour = 0
        state = "AM"
    print(f"{self.hour:02d}:{self.minute:02d}")

def main():
    clock = Clock(12, 50, 12)
    clock.set_time(9,30,24)
    clock.set_hour(13)
    clock.set_seconds(30)
    clock.get_time()

main()

```

The terminal on the right shows the execution of the script using PowerShell:

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMITL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No1.py"
01:30:30 PM
PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>

```

The status bar at the bottom indicates the file is at line 50, column 31, with 4 spaces, UTF-8 encoding, CRLF line endings, and is running Python 3.10.5 64-bit.

No.2

```
class Poly:
    def __init__(self, x):
        self.x = list(x)

    def who_is_bigger(self, p):
        first_power = self.find_powerOfx()
        first_coef = self.x

        second_power = p.find_powerOfx()
        second_coef = p.x

        if (len(first_power) >= len(second_power)):

            bigger_coef = first_coef
            bigger_power = first_power

            smaller_coef = second_coef
            smaller_power = second_power

        elif (len(first_power) < len(second_power)):

            bigger_coef = second_coef
            bigger_power = second_power

            smaller_coef = first_coef
            smaller_power = first_power

        while(len(smaller_power) != len(bigger_power)):
            smaller_power += (0,)
            smaller_coef += (0,)

        return (smaller_coef, smaller_power, bigger_coef, bigger_power)

    def add(self, p):
        (smaller_coef, smaller_power, bigger_coef, bigger_power) = self.who_is_bigger(p)

        new_coef = []
        for i in range(0, len(bigger_power)):

            if bigger_power[i] == smaller_power[i]:
                new_coef += (smaller_coef[i] + bigger_coef[i],)

            else:
                new_coef += (bigger_coef[i],)

        self.x = new_coef
        return Poly(self.x)

    def scalar_multiply(self, n):
```

```

        for i in range(len(self.x)):
            self.x[i] *= n

        return Poly(self.x)

def multiply(self, p):
    #(smaller_coef, smaller_power, bigger_coef, bigger_power) = self.who_is_bigger(p)

    new_coef = []

    # print(len(self.x) + len(p.x) - 1)
    for i in range(0, (len(self.x) + len(p.x)) - 1 ):
        new_coef += (0, )

        # [0 + 0] += 1*1
        # [0 + 1] += 1*1
        # [1 + 0] += 1*1
        # [1 + 1] += 1*1
        # [2 + 0] += 1*1
        # [2 + 1] += 1*1
        # print(f"new_coef = {new_coef}")

        i = 0
        while(i != len(self.x)):
            j = 0
            while(j != len(p.x)):
                new_coef[i+j] += (self.x[i] * p.x[j])
                j += 1
            i += 1

        # print(f"new_coef = {new_coef}")

    return Poly(new_coef)

def power(self, n):

    new_coef = []

    # print(len(self.x) + len(self.x))
    for i in range(0, (len(self.x) * n ) - 1 ):
        new_coef += (0, )

        # [0 + 0] += 1*1
        # [0 + 1] += 1*1
        # [1 + 0] += 1*1
        # [1 + 1] += 1*1
        # [2 + 0] += 1*1
        # [2 + 1] += 1*1
        # print(f"new_coef = {new_coef}")

        i = 0
        while(i != len(self.x)):
            j = 0
            while(j != len(self.x)):
                new_coef[i+j] += (self.x[i] * self.x[j])
                j += 1

```

```

        i += 1

    # print(f"new_coef = {new_coef}")

    return Poly(new_coef)

def find_powerOfx(self):
    count = 0
    powers = ()
    for i in self.x:

        if (i == 0):
            powers += (0,)
            count += 1
            continue
        else:
            powers += (count,)
            count += 1

    return powers

def diff(self):

    new_coef = ()

    for i in range(len(self.x)-1):

        new_coef += (self.x[i + 1] * (i + 1),)

    return Poly(new_coef)

def integrate(self):
    #  $\int x^n dx = (x^{(n+1)} / (n+1)) + C ; n \neq 1$ 

    new_coef = []
    # print(f"len(self.x) = {len(self.x)}")
    for i in range(0, len(self.x)+1):
        new_coef += (0, )

    # print(f"new_coef = {new_coef}")
    original_powers = self.find_powerOfx()

    print()

    for i in range(0, len(self.x)):
        # print(f"self.x[i] / original_powers[i] + 1 = {self.x[i] // original_powers[i] + 1}")

        new_coef[i+1] = (self.x[i] / (original_powers[i] + 1))

    # print(f"self.x[i] : {self.x[i]}")

```

```

        # print(f"original_powers[i] : {original_powers[i] + 1}")
        # print()

    # print(f"new_coef = {new_coef}")

    return Poly(new_coef)

def print(self):
    count = 0

    for i in self.x:
        sign = "+"
        if i < 0:
            sign = "-"
        if (count == 0):
            sign = ""
        if (i == 0):
            count += 1
            continue
        else:
            i = abs(i)
            if count == 0:
                print(f"{sign}{int(i)} ", end = "")
            else:
                print(f"{sign} {int(i)}x^{count} ", end = " ")
            count += 1
    print()

def eval(self, n):
    count = 0
    total = 0
    for i in self.x:

        if (i == 0):

            # print(f"Skipped {count}^th power since it's coeff is 0.")
            count += 1
            continue
        else:
            # print(f"total = {total}")
            total += i * (n ** count)
            # print(f"{i} * ({n} ** {count}) = {total}")
            count += 1

    print(total)

def main():
    #          0th, 1th, 2th, 3th, 4th, ....
    p = Poly( ( 1, 2, 3) )
    p.print()

    print()
    p.diff().print()
    p.integrate().print()

    # q = p.power(2)

```

```
# q.print()
# p.print()

# p.eval(3)
# print(p.find_powerOfx())
# print(q.find_powerOfx())

# r = p.add(q)
# r.print()

# p.scalar_multiply(2)
# p.print()

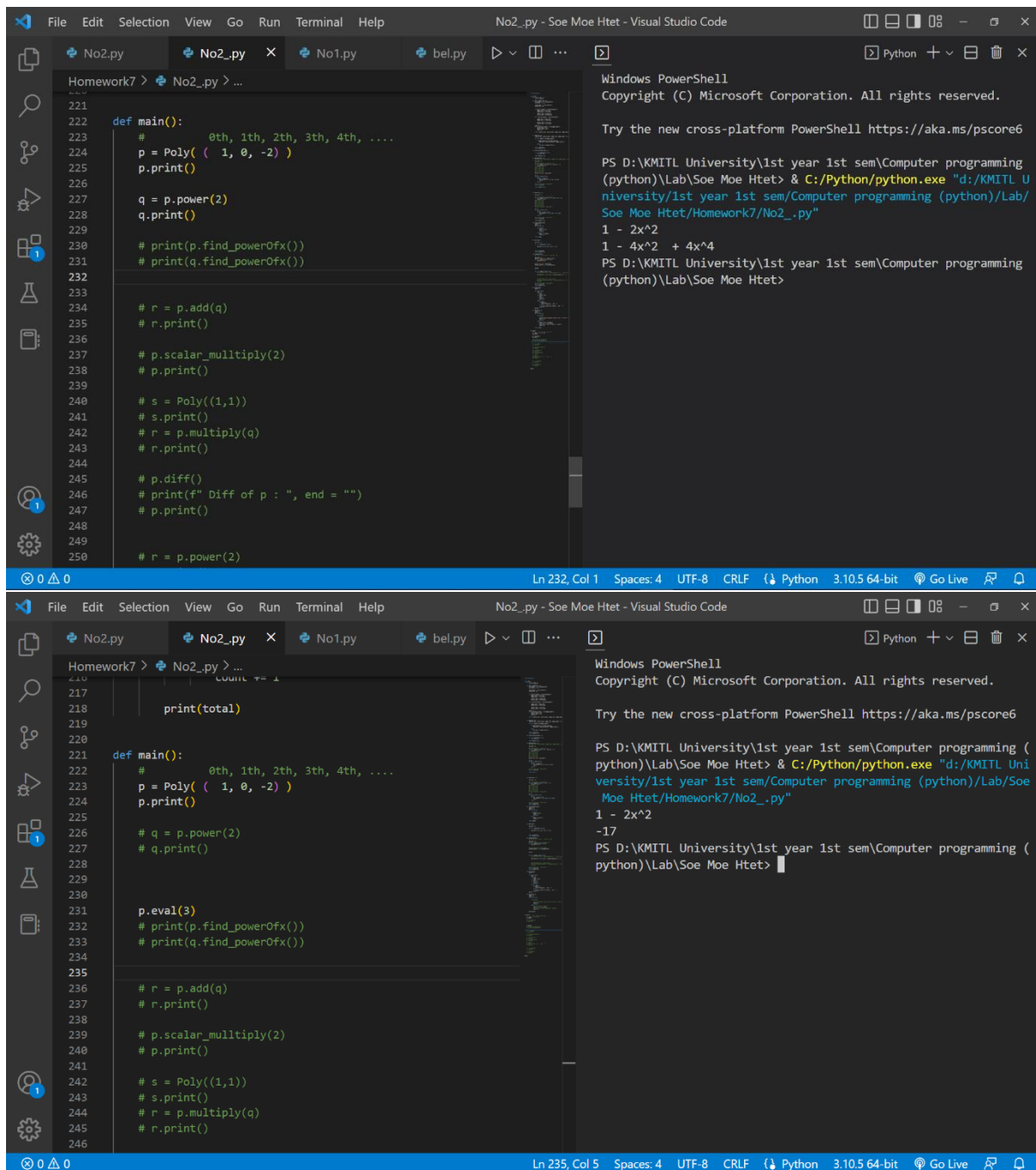
# s = Poly((1,1))
# s.print()
# r = p.multiply(s)
# r.print()

# p.diff()
# print(f"Diff of p : ", end = "")
# p.print()

# r = p.power(2)
# r.print()
```

```
main()
```


Sample output



The image displays two screenshots of a Visual Studio Code editor window, showing a Python script and its execution output in a PowerShell terminal.

Top Screenshot:

- The editor shows a file named `No2.py` with the following Python code (lines 221-250):

```
221
222 def main():
223     # 0th, 1th, 2th, 3th, 4th, ....
224     p = Poly( ( 1, 0, -2 ) )
225     p.print()
226
227     q = p.power(2)
228     q.print()
229
230     # print(p.find_powerOfx())
231     # print(q.find_powerOfx())
232
233
234     # r = p.add(q)
235     # r.print()
236
237     # p.scalar_multiply(2)
238     # p.print()
239
240     # s = Poly((1,1))
241     # s.print()
242     # r = p.multiply(q)
243     # r.print()
244
245     # p.diff()
246     # print(f" Diff of p : ", end = "")
247     # p.print()
248
249
250     # r = p.power(2)
```
- The terminal shows the output of the script execution:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMITL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"
1 - 2x^2
1 - 4x^2 + 4x^4
PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>
```

Bottom Screenshot:

- The editor shows the same Python script, but with additional lines (lines 217-219) added before line 221:

```
217
218     print(total)
219
220
221 def main():
```
- The terminal shows the output of the script execution, including the new lines:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMITL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"
1 - 2x^2
-17
PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>
```

The screenshot shows the Visual Studio Code editor with a Python file named `No2.py` open. The code defines a polynomial `p` and performs various operations including evaluation, finding power of x , addition, scalar multiplication, and multiplication. The PowerShell terminal on the right shows the execution of the script, displaying the polynomial $1 - 2x^2$ and its powers at $x=3$.

```
def main():
    # 0th, 1th, 2th, 3th, 4th, ....
    p = Poly( ( 1, 0, -2 ) )
    p.print()

    q = p.power(2)
    q.print()
    p.print()

    # p.eval(3)
    # print(p.find_powerOfx())
    # print(q.find_powerOfx())

    r = p.add(q)
    r.print()

    # p.scalar_multiply(2)
    # p.print()

    # s = Poly((1,1))
    # s.print()
    # r = p.multiply(q)
    # r.print()
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS D:\KMIL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMIL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"
1 - 2x^2
1 - 4x^2 + 4x^4
1 - 2x^2
2 - 6x^2 + 4x^4
PS D:\KMIL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>

This screenshot shows the same Python script as above, but with additional operations: scalar multiplication by 2, differentiation, and printing the difference of the polynomial. The PowerShell terminal shows the updated output, including the scalar multiplication result $2 - 4x^2$ and the derivative $2 - 4x^2$.

```
def main():
    # 0th, 1th, 2th, 3th, 4th, ....
    p = Poly( ( 1, 0, -2 ) )
    p.print()

    # q = p.power(2)
    # q.print()
    # p.print()

    # p.eval(3)
    # print(p.find_powerOfx())
    # print(q.find_powerOfx())

    r = p.add(q)
    r.print()

    p.scalar_multiply(2)
    p.print()

    # s = Poly((1,1))
    # s.print()
    # r = p.multiply(q)
    # r.print()

    # p.diff()
    # print(f" Diff of p : ", end = "")
    # p.print()
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell <https://aka.ms/pscore6>
PS D:\KMIL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMIL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"
1 - 2x^2
1 - 4x^2 + 4x^4
1 - 2x^2
2 - 6x^2 + 4x^4
PS D:\KMIL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMIL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"
1 - 2x^2
2 - 4x^2
PS D:\KMIL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>

The screenshot shows the Visual Studio Code editor with a file named `No2.py` open. The code defines a `main` function that performs polynomial operations using the `sympy` library. The terminal on the right shows the output of the script, which includes the polynomial $1 - 2x^2$ and its derivative $-4x$.

```
def main():
    # 0th, 1th, 2th, 3th, 4th, ....
    p = Poly( 1, 0, -2 )
    p.print()

    # q = p.power(2)
    # q.print()
    # p.print()

    # p.eval(3)
    # print(p.find_powerOfx())
    # print(q.find_powerOfx())

    # r = p.add(q)
    # r.print()

    # p.scalar_multiply(2)
    # p.print()

    s = Poly((1,1))
    s.print()
    r = p.multiply(s)
    r.print()

    p.diff()
    print(f"Diff of p : ", end = "")
    p.print()
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMITL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"

```
1 - 2x^2
1 + 1x^1
1 + 1x^1 - 2x^2 - 2x^3
PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>
```

The screenshot shows the Visual Studio Code editor with a file named `No2.py` open. The code defines a `main` function that performs polynomial operations using the `sympy` library. The terminal on the right shows the output of the script, which includes the polynomial $1 - 2x^2$ and its derivative $-4x$.

```
def main():
    # 0th, 1th, 2th, 3th, 4th, ....
    p = Poly( 1, 0, -2 )
    p.print()

    # q = p.power(2)
    # q.print()
    # p.print()

    # p.eval(3)
    # print(p.find_powerOfx())
    # print(q.find_powerOfx())

    # r = p.add(q)
    # r.print()

    # p.scalar_multiply(2)
    # p.print()

    s = Poly((1,1))
    s.print()
    r = p.multiply(s)
    r.print()

    p.diff()
    print(f"Diff of p : ", end = "")
    p.print()
```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMITL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"

```
1 - 2x^2
1 + 1x^1
1 + 1x^1 - 2x^2 - 2x^3
PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>
Diff of p : - 4x^1
PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>
```

The image shows a Visual Studio Code window with the title "No2_py - Soe Moe Htet - Visual Studio Code". The editor has three tabs: "No2.py", "No2.py", and "No1.py". The active tab is "No2.py", which contains a Python script. The script starts with a loop that prints powers of 2, then a function `main()` that uses the `Poly` class to print, differentiate, and integrate a polynomial, and finally evaluates the polynomial at `x=3`.

```
206
207
208     # print(f'Skipped {count}^th power since it's co
209     count += 1
210     continue
211 else:
212     # print(f'total = {total}')
213     total += i * (n ** count)
214     # print(f'{i} * {n} ** {count} = {total}')
215     count += 1
216
217     print(total)
218
219 def main():
220     #      0th, 1th, 2th, 3th, 4th, ....
221     p = Poly( ( 1, 2, 3) )
222     p.print()
223
224     print()
225     p.diff().print()
226     p.integrate().print()
227
228
229     # q = p.power(2)
230     # q.print()
231     # p.print()
232
233     # p.eval(3)
234     # print(p.find_powerOfx())
235     # print(q.find_powerOfx())
236
```

The PowerShell terminal on the right shows the output of the script. It displays the polynomial $1 + 2x^1 + 3x^2$, its derivative $2 + 6x^1$, and its integral $+ 1x^1 + 1x^2 + 1x^3$. The terminal also shows the command used to run the script: `PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMITL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"`.

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet> & C:/Python/python.exe "d:/KMITL University/1st year 1st sem/Computer programming (python)/Lab/Soe Moe Htet/Homework7/No2_.py"

1 + 2x^1 + 3x^2

2 + 6x^1

+ 1x^1 + 1x^2 + 1x^3

PS D:\KMITL University\1st year 1st sem\Computer programming (python)\Lab\Soe Moe Htet>

Ln 223, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.5 64-bit Go Live

No3

```
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 get_a(self, a):
        return self.__a

    def get_b(self, b):
        return self.__b

    def get_c(self, c):
        return self.__c

    def get_d(self, d):
        return self.__d

    def get_e(self, e):
        return self.__e

    def get_f(self, f):
        return self.__f

    def isSolvable(self):
        if( (self.__a * self.__d) - (self.__b * self.__c) != 0 ):
            return True
        else:
            return False

    def getX(self):
        top = ( self.__e * self.__d) - (self.__b * self.__f)
        bottom = (self.__a * self.__d) - (self.__b * self.__c)
        return top / bottom

    def getY(self):
        top = ( self.__a * self.__f) - (self.__e * self.__c)
        bottom = (self.__a * self.__d) - (self.__b * self.__c)
        return top / bottom
```