1 (20)	2 (20)	3 (20)	4 (20)	5 (20)	TOTAL (100)

[CS101] Introduction to Programming 2016 Spring - Midterm Examination

SECTION	STUDENT ID	NAME

※ Please check if you have all 20 pages of the test material.
※ 시작하기 전에 반드시 페이지의 수를 확인 하십시오.(전체 : 20쪽)

※ Fill in your student identification number and name. Otherwise you will lose 1 point for each missing piece of information.
※ 위의 정보(학번,이름)를 정확히 기입하지 않을 경우, 각 실수 당 1점이 감점 됩니다.

※ **TAs will not answer your questions about the exam**. If you think that there is anything ambiguous, unclear or wrong about a problem, please write the reasons and make necessary assumptions to solve the problem. We will take your explanation into consideration while grading.
※ **시험시간동안 질문을 받지 않습니다**. 만일 문제에 오류나 문제가 있을 경우, 왜 문제가 이상이 있다고 생각하는지에 대해서 기술하시면 되겠습니다. 또한 문제가 애매하다고 생각되는 경우 문제를 푸실 때 본인이 생각하는 가정을 함께 작성하셔서 문제를 푸시면 되겠습니다. 채점 시 가정 및 설명을 고려하도록 하겠습니다.

※ **Stick to Python 3**. We will grade answers only in Python 3.
※ **파이썬 3만 사용하십시오**. 채점은 파이썬 3 기준으로만 합니다.

※ Checking and claiming on the exam scores are handled at Lab 6 of individual section.
※ 시험 점수 확인 및 클레임은 자신이 속한 분반의 Lab 6 에서 진행됩니다.

1-1. (5 points/1 point each) What will the last statement output? Answer "True" or "False". The code in (d) will output two answers.

Statement		Answer

(a)	`x = (100, 200, 300)`
	`y = x`
	`print ( x is y )`	

(b)	`x = (100, 200, 300)`
	`y = (100, 200, 300)`
	`print ( x is y )`	

(c)	`x = (100, 200, 300)`
	`y = (100, 200, 300)`
	`print ( x == y )`	

(d)	`x = (100, 1000)`
	`y = ("100", 1000)`
	`for i in x:`
	`    print ( i in y )`	

1-2. (4 points) When the following four statements are executed, they will output an error. Match the statements with the types of error listed below.
(※ **Note: Statements (a) to (d) and errors (1) to (4) are 1:1 mapped.**)

Statement		Answer

(a)	`x = ("CS101", "A+", "Happy")`
	`"CS101" in y`	

(b)	`x = "100"`
	`y = (200, 300)`
	`z = x + y`	

(c)	`x = (1,2,3,4,5]`	

(d)	`x = 10 % 5`
	`y = 10 ** 5`
	`y // x`	

(1) SyntaxError   (2) TypeError   (3) NameError   (4) ZeroDivisionError

1-3. (11 points) What will the last statement in the follow pieces of code print?

	Statement	Answer
(1)	`print ( 11/5 )`	(1 point)
(2)	`print ( 11//5 )`	(1 point)
(3)	`print ( ("Hello" + "!!")*2 )`	(1 points)

(4)	`def set_x(y):`
	`  global x`
	`  x = y`
	`print ( set_x(0) )`
		(2 points)

(5)	`def id():`
	`  global x`
	`  y = x`
	`  return y`
	`x = ('Kildong', '19980101')`
	`y = id()`
	`print (x is y)`
		(2 points)

(6)	`x = (10, 20, 30)`
	`y = x`
	`x = (100, 200, 300)`
	`print ( y )`
		(2 points)

(7)	`x = 7`
	`y = 7.0`
	`print ( x == y and not y == 7 )`
		(2 points)

(8)	`def set_x():`
	`  global z`
	`  z = 1000`
	`  return True`
	`x = 1`
	`y = 10`
	`z = 100`
	`x == y and set_x()`
	`print ( z )`
		(2 points)

2. (20 points) In this problem we start with the code from homework 2-2 given below.

Original code

Column 1

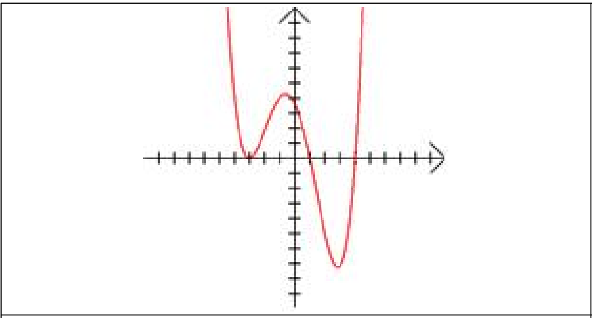
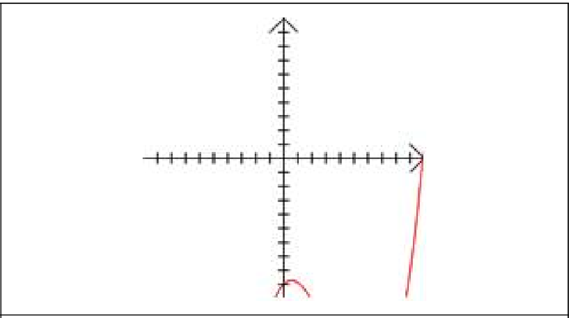
```
from cs1graphics import *
Graph = Canvas(500, 500)
# Draw the x and y axes
def draw_axes():
  x_axis = Path(Point(0, 250),
                Point(500, 250))
  x_ar = Path(Point(475, 225),
              Point(500, 250),
              Point(475, 275))
  Graph.add(x_axis)
  Graph.add(x_ar)
  y_axis = Path(Point(250, 0),
               Point(250, 500))
  y_ar = Path(Point(225, 25),
              Point(250, 0),
              Point(275, 25))
  Graph.add(y_axis)
  Graph.add(y_ar)
  for i in range(20):
    x_tick=Path(Point(25*(i+1),240),
                Point(25*(i+1),260))
    y_tick=Path(Point(240,25*(i+1)),
                Point(260,25*(i+1)))
    Graph.add(x_tick)
    Graph.add(y_tick)
```

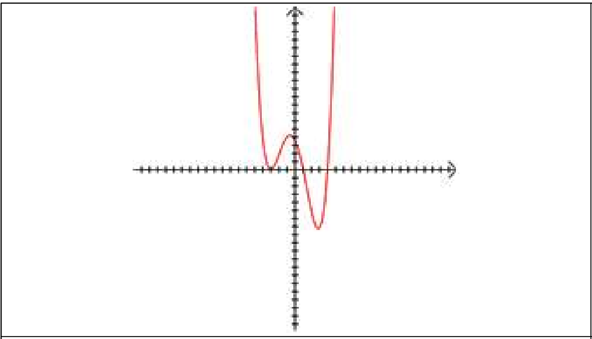
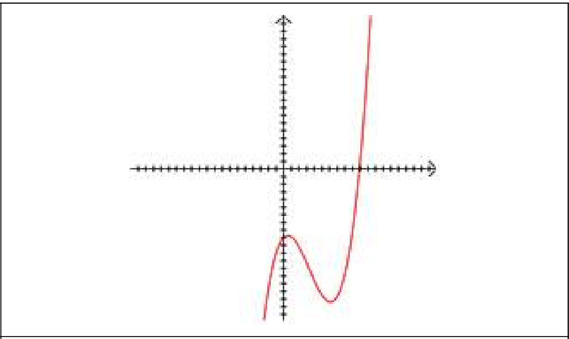
Column 2

```

# Draw the graph 'f(x) = ax^4 + bx^3
+ cx^2 + dx + e'
def draw_graph(a, b, c, d, e):
  graph = Path()
        Problem 2-3
  graph.setBorderColor("red")
  Graph.add(graph)

# Run program entry
def main():
  print("f(x) = ax^4 + bx^3 + cx^2 +
dx + e")
  a = float(input("Insert a : "))
  b = float(input("Insert b : "))
  c = float(input("Insert c : "))
  d = float(input("Insert d : "))
  e = float(input("Insert e : "))

  draw_axes()
  draw_graph(a, b, c, d, e)

main()
```

In Figure (1) below, two graphs from equations (a) and (b) are drawn from executing the code above. However, the knees of equation (b) do not lie in the fixed canvas size of 500x500. If we increase the canvas size to 1000x1000, we can see the knees of equation (b) as in Figure (2).
Assume that your pair programming partner modified the code in Columns 1 and 2 into that in Columns 3 and 4 but did not complete it. Fix the code as instructed.

Note:
1. The tick interval on both x and y axes in the figures is fixed at 25 pixels.
2. Do not use the **`import`** statement.

	
(a) $f(x) = 0.1x^4 + 0.1x^3 - 1.7x^2 - 2.1x + 3.6$	(b) $f(x) = 0x^4 + 0.1x^3 - 1x^2 + 1x - 9$
Figure (1): The canvas size is 500x500.	

	
(a) $f(x) = 0.1x^4 + 0.1x^3 - 1.7x^2 - 2.1x + 3.6$	(b) $f(x) = 0x^4 + 0.1x^3 - 1x^2 + 1x - 9$
Figure (2): The canvas size is 1000x1000.	

Revised code

Column 3

```
from cs1graphics import *
g_w = 500 # Canvas width
g_h = 500 # Canvas height
intv = 25 # tick interval
Graph = None
# Draw the x and y axes
def draw_axes():
    Problem 2-2
```

Column 4

```
# Draw the graph 'f(x) = ax^4 + bx^3
+ cx^2 + dx + e'
def draw_graph(a, b, c, d, e):
    Problem 2-3
# Program entry function
def main():
    Problem 2-1
main()
```

2-1. (4 points) Assume that your pair programming partner modified the **`main`** function as in Column 5 below. The code asks a user to input the size of the canvas. However, when you run the program, you get the following error.

```
AttributeError: 'NoneType' object has no attribute 'add'
```

You should add code to fix the error and specify the line number where you add the code.

Line number	**Column 5**
1	`def main():`
2	
3	`  g_w = int(input("Width of the graph: "))`
4	
5	`  g_h = int(input("Height of the graph: "))`
6	
7	`  Graph = Canvas(g_w, g_h)`
8	
9	`  draw_axes()`
10	
11	`  print("f(x) = ax^4 + bx^3 + cx^2 + dx + e")`
12	
13	`  a = float(input("Insert a : "))`
14	
15	`  b = float(input("Insert b : "))`
16	
17	`  c = float(input("Insert c : "))`
18	
19	`  d = float(input("Insert d : "))`
20	
21	`  e = float(input("Insert e : "))`
22	
23	`  draw_axes()`
24	
25	`  draw_graph(a, b, c, d, e)`

Answer)

2-1	Line number: Code:

2-2. (8 points) Your pair programming partner modified the **`draw_axes`** function as below, and it works fine for the most part. But, sometimes, the function draws a **weird x-axis that has insufficient ticks**. Describe when and why such a problem takes place and write down the modified code to fix it. Assume that the user inputs only **g_w** and **g_h** such that 100 < g_w, g_h < 10000. Hint) Think about cases that depend on the user's input.

```
def draw_axes():
  x_axis = Path(Point(0, g_h / 2), Point(g_w, g_h / 2))
  x_ar = Path(Point(g_w-5,g_h/2−5),Point(g_w,g_h/2),Point(g_w-5,g_h/2+5))
  Graph.add(x_axis)
  Graph.add(x_ar)
  y_axis = Path(Point(g_w / 2, 0), Point(g_w / 2, g_h))
  y_ar = Path(Point(g_w/2-5,5), Point(g_w/2,0), Point(g_w/2+5,5))
  Graph.add(y_axis)
  Graph.add(y_ar)
  for i in range(int(g_h / intv)):
    y_tick = Path(Point(g_w/2-10,intv*(i+1)), Point(g_w/2+10,intv*(i+1)))
    x_tick = Path(Point(intv*(i+1),g_h/2-10), Point(intv*(i+1),g_h/2+10))
    Graph.add(y_tick)
    Graph.add(x_tick)
```

Answer)

Problem	
Modified Code	

2-3. (8 points) Complete the **`draw_graph`** function. You should use **`g_w`**, **`g_h`** and **`intv`** variables for drawing $f(x)$ graph in a canvas of any size.

```
def draw_graph(a, b, c, d, e):
  graph = Path()

    Problem 2-3

  graph.setBorderColor("red")
  Graph.add(graph)
```

Hint) The **`addPoint`** function of **`Path`** appends a new point to the end of the current sequence of points. In the example below, **`path1`** and **`path2`** are of the same shape.

```
path1 = Path(Point(0, 250), Point(500, 250))
path2 = Path()
path2.addPoint(Point(0, 250))
path2.addPoint(Point(500, 250))
```

Answer)

2-3	

3. (20 points) Answer each question according to the instruction.

3-1. (7 points) What is the result of the following program?

3-1-1 (3 points)	3-1-2 (4 points)

`def absolute(x):`
`    if x < 0:`
`        return -x`
`    if x > 0:`
`        return x`
`print(absolute(0))`
	`a = 13`
	`b = 3.1415`
	`c = True`
	`d = 3+6j`
	`print(type(a))`
	`print(type(input('enter a number:')))`
	`# Assume you enter a number 2`
	`print(type(c))`
	`print(type(d))`

(3-1-1)	
(3-1-2)	______ enter a number:2 ______ ______ ______

3-2. (4 points) What is the outcome of the following program?

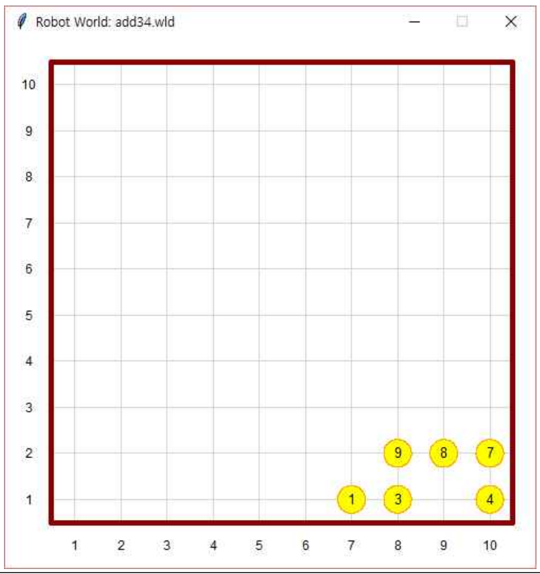
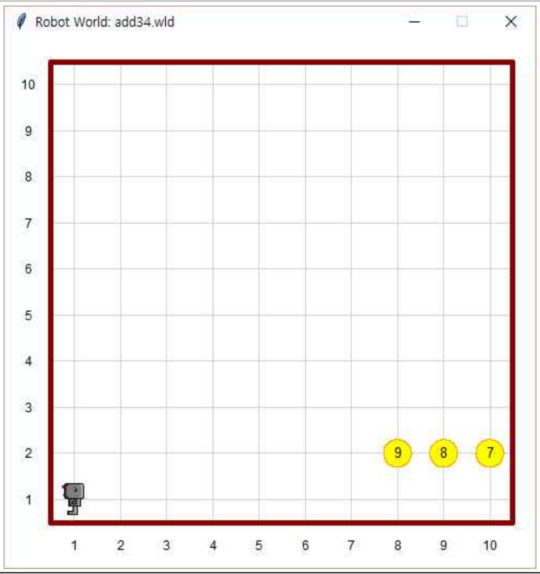
```
for i in range(3):
    string = ""
    for j in range(i):
        string +="X"
        for k in range(j * 3):
            string += "O"
    print(string)
```


3-3. (4 points) What is the outcome of the following program?

```
if True and False or True:
    print('“Hello, professor”')
if False and True or True:
    print('“CS101 is fantastic”')
else:
    print('CS101 is not fantastic')
```


3-4. (5 points) Write down the **pickBeeper** function such that we see the screen output and terminal output on page 12 after we execute the code below.

```
from cs1robots import *
load_world('worlds/add34.wld')
hubo = Robot()
def pickBeeper():
    #3-4 ( Complete pickBeeper function
            that will be used inside readNumber function. )
def init():
    while not hubo.facing_north():
        hubo.turn_left()
    hubo.turn_left()
    hubo.turn_left()
    hubo.turn_left()
    while hubo.front_is_clear():
        hubo.move()
    hubo.turn_left()
    hubo.turn_left()
def readNumber():
    init()
    num = 0
    base_digit = 0
    while hubo.front_is_clear():
        num = num + pickBeeper()*10**base_digit
        base_digit = base_digit + 1
        hubo.move()
    return num
print(readNumber())
```

add34.wld	SCREEN OUTPUT (After the program)
	
	TERMINAL
	1304

(3-4)	

4. (20 points) Answer each question following the instructions.

4-1. (8 points) Implement the following functions.

$$f_1(x,y,z) = 1 + 2f_2(x,y) - 3f_3(x,y,z)$$
$$f_2(x,y) = 4x - 5f_4(x,y) + 6y$$
$$f_3(x,y,z) = 7x^2 + 8xy + 9z^2$$
$$f_4(x,y) = 10x + 11y$$

```
def function1(x,y,z):
```

4-1-1

```
def function2(x,y):
```

4-1-2

```
def function3(x,y,z):
```

4-1-3

```
def function4(x,y):
```

4-1-4

4-2. (12 points) Solving quadratic equations.

A quadratic equation has the form, $ax^2+bx+c=0$, where *x* is the unknown and *a, b,* and *c* are known coefficients. Real values of *x* that satisfy the equation are called the *roots* and can be algebraically found using the *quadratic formula* shown below.

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

The process of finding the roots is called "solving" the quadratic equation. An important step in solving the quadratic equation is the discriminant analysis. The *discriminant* is given by the term inside the square root, and it is a key indicator for the existence of the solution. A quadratic equation can be solved if and only if the discriminant is larger or equal to zero. The rules of solving the quadratic equation using the discriminant are outlined below:

Let *D* denote the discriminant, $D=b^2-4ac$,

(1) if $D < 0$, no real roots exist

(2) if $D = 0$, a unique root is given by the simplified quadratic equation, $x=\frac{-b}{2a}$

(3) if $D > 0$, the two roots are given by the quadratic equation.

Write a program that receives user inputs for the coefficients *a*, *b*, and *c* separately and outputs the root(s) of the quadratic equation. If the program cannot solve the quadratic equation specified the user, the program should print an appropriate message.

Note. Your program can print floating point answers. It does not have to print in fraction forms. That is, *x* = 0.5 is okay. You don't have to print *x* = 1/2.

Note. A function **`math.sqrt(x)`** has been imported and provided for you. It receives a number as input in place of the parameter x, and returns the square root of *x* in float.
Ex)

```
>>> sqrt(9)
3.0
```

Some examples of the expected interactions are shown below.

```
input a: 1
input b: 1
input c: 1
Discriminant is smaller than zero! No Real Roots!
```

```
input a: 1
input b: 2
input c: 1
A unique root exists:
x = 0.5
```

```
input a: 2
input b: 5
input c: -3
Real roots exist:
x = -3
x = 0.5
```

Solution:

```
from math import sqrt
```

5. (20 points) Answer each question according to the instruction.

5-1. (8 points) What is the outcome of the following program?

```
def func1(var2, var3):
   var2 += 1
   var3 = 0

def func2(var1):
   global var2
   var2 = 9
   var1 += 7
   return var1

def func3(var):
   var2 = str(var)
   var2 *= 3

var1, var2, var3, var4 = 1, 0, 0, 4
func1(var2, var3)
print(var1, var2, var3, var4)

var1, var2, var3, var4 = 1, 0, 0, 4
var1 = func2(var3)
var3 +=2
(var3, var4) = (var4, var3)
print(var1, var2, var3, var4)

var1, var2, var3, var4 = 1, 0, 0, 4
var2 = func3(0)
print(var1, var2, var3, var4)
```

Here are the descriptions of the objects and their methods used in (5-2) and (5-3).

Function	Description
Canvas(width, height, ‘backgroundColor’, ‘title’)	Create a new canvas for drawing.
canvas.add(object)	Add a Drawable object to the canvas.
Square(length of a side, centerPoint(x, y))	Return a new Square object.
Circle(radius, centerPoint(x, y))	Return a new Circle object.
sqrt(number)	Return the square root of the number.
object.setDepth(depth)	Set the depth of the object. ※ Note: Objects with a smaller depth appear in foreground.
object.setFillColor(‘color’)	Set the interior color of the object to the given ‘color’
object.rotate(angle in a degree)	Rotate the object around its center point.
object.scale(scaling factor)	Make an object smaller or larger with scaling factor
object.clone()	Return a duplicate of the drawable object.
object.move(dx, dy)	Move the object dx units along X-axis and dy units along Y-axis.

5-2 (2 points) Paint appropriate pixels black in “<answer for img>” according to the outcome of the program below.

```
from cs1media import *

img = create_picture(5,5,Color.white)
w, h = img.size()
for y in range(h) :
    for x in range(w) :
        if (x == y) or (not (x * y != 0)) :
            img.set(x, y, Color.black)
img.show()
```

<answer for img>

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)

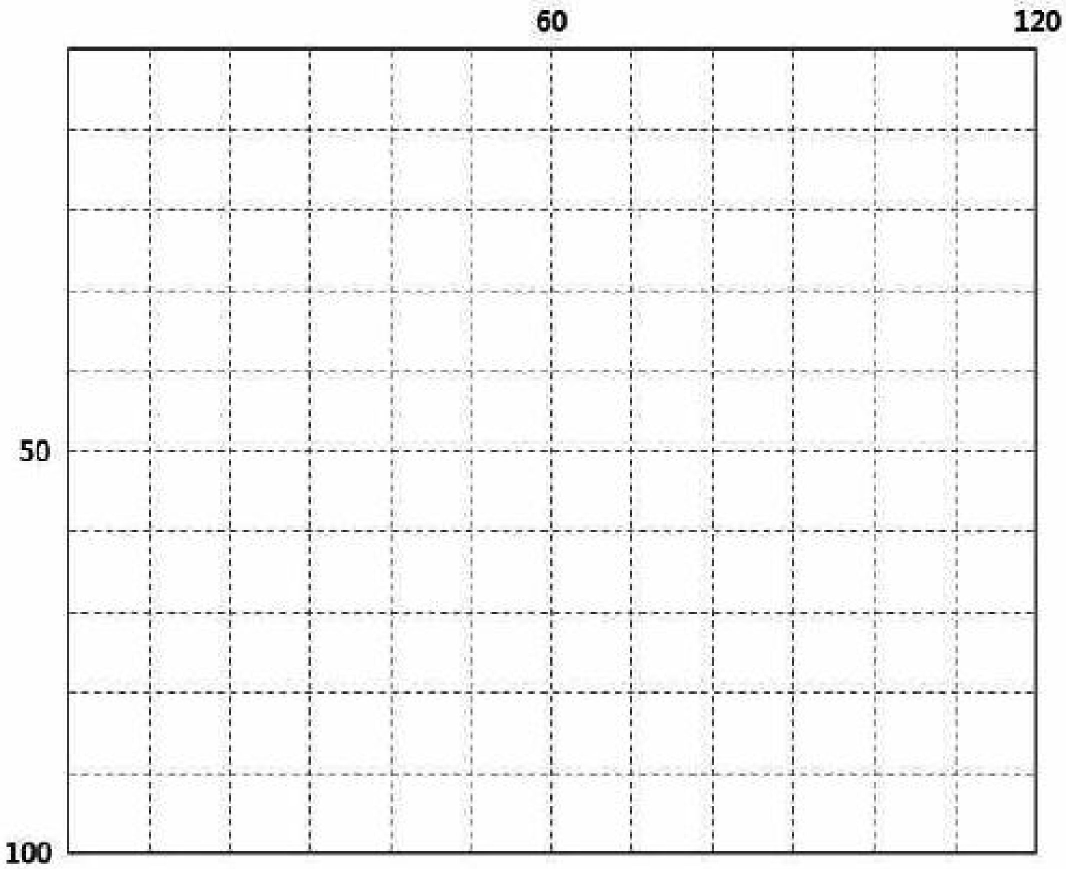
5-3 (2 points) What is the outcome of the following program?

```
from cs1graphics import *

book = Square(50)
book.setFillColor("Red")
desk = book
desk.setFillColor("Green")
photo = book
photo.setFillColor("Yellow")
print(desk.getFillColor())
```

____

5-4. (8 points) What is the outcome of following program? Draw it on the canvas.

```
from cs1graphics import *
import math

paper = Canvas( 120, 100, 'white', 'Canvas' )
sq1 = Square( 50, Point( 35,50 ) )
sq1.setDepth( 50 )
sq1.setFillColor( 'black' )
paper.add( sq1 )

sq2 = sq1.clone()
sq2.setFillColor( 'white' )
sq2.rotate( 45 )
sq2.setDepth( 25 )
sq2.scale( math.sqrt(2) / 2 )
paper.add( sq2 )

sq3 = sq2.clone()
sq3.move( 50, 0 )
paper.add( sq3 )

circle = Circle( 25, Point( 85, 50 ) )
circle.setFillColor( 'black' )
circle.setDepth( 75 )
paper.add( circle )
```


60
120
50
100

<Canvas>