

Question:	1	2	3	4	5	Total
Points:	20	20	20	20	20	100
Score:						

**KAIST CS101 Introduction to Programming**  
**2017 Spring Midterm Exam**

**Section:** \_\_\_\_\_



**Student ID:** \_\_\_\_\_

**Name:** \_\_\_\_\_

- Please check if you received all 21 pages of the test material.  
시작하기 전에 반드시 페이지가 총 21쪽인지 확인 하십시오.
- Fill in your student identification number and name. Otherwise you will lose 1 point for each missing piece of information.  
학번과 이름을 정확히 기입하지 않을 경우, 각 실수 당 1점이 감점됩니다.
- **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 이상이 있을 경우, 왜 문제가 이상이 있다고 생각하는지에 대해 기술하시면 됩니다. 문제가 애매하다고 생각되는 경우 문제를 풀 때 본인이 생각하는 가정을 함께 작성하면 됩니다. 채점 시 가정 및 설명을 고려하겠습니다.
- **Write your answer in Python 3.** We will grade your answers based only on Python 3.  
**Python 3를 사용해 문제를 해결하세요.** 채점은 Python 3 기준으로만 진행됩니다.

1. (20 points) Answer the following questions.

1-1. (6 points) What will be the outputs of the following Python code snippets?

Code	Output (or Error)
<pre>print(False == False or True)</pre>	
<pre>print( 7 / 3 - 7 // 3 * 7 % 3)</pre>	
<pre>l1 = [1,2] l2 = [1,2] print(l1 is l2)</pre>	
<pre>t = 5,4,3 t[1] = t[2] - 3 print(t[t[1]])</pre>	
<pre>l1 = [1,2,3] l2 = l1[:-2] l2.append(4) print(sum(l1))</pre>	
<pre>l1 = [1,3,5] print(l1.index(2))</pre>	

1-2. (2 points) What will be the output of the following Python code?

```

1 def swap(a, b):
2     a, b = b, a
3
4 a, b = 1, 2
5 swap(a, b)
6
7 print (a)
```

**[Answer box]**

1-3. (2 points) What will be the output of the following Python code?

```
1 x = 10
2 y = 0
3
4 def add_inc_x(y):
5     global x
6     x = x + 1
7     return x + y
8
9 if (x < add_inc_x(y) or x > add_inc_x(y)) :
10     print(x)
11 else:
12     print(y)
```

[Answer box]

1-4. (5 points) What will be the output of the following Python code?

```
1 def nand(a, b):
2     return not (a and b)
3
4 t = True
5 f = False
6
7 print( (t or t) == nand(nand(t,t), nand(t,t)) )
8 print( (t or f) == nand(nand(t,t), nand(f,f)) )
9 print( (f or t) == nand(nand(f,f), nand(t,t)) )
10 print( (f or f) == nand(nand(f,f), nand(f,f)) )
11 print( (t or f) == nand(nand(t,f), nand(t,f)) )
```

[Answer box]

1-5. (5 points) What will be the output of the following Python code?

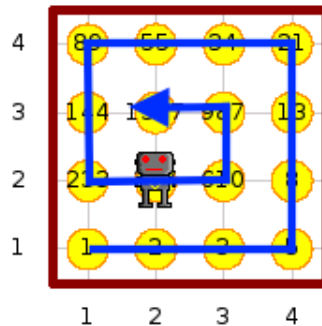
```
1 l1 = [ 3, 5, 2, 4]
2 for i in range(len(l1)):
3     for j in range(len(l1)-i-1):
4         if l1[j] > l1[j+1]:
5             l1[j], l1[j+1] = l1[j+1], l1[j]
6 print(l1)
```

**[Answer box]**

2. (20 points) This question is related to your homework assignments. However, make no assumptions about available homework specific functions unless stated in the question. Read the instructions carefully and answer each question according to the instruction.

2-1. You need to move hubo in a clockwise snail pattern as shown below while dropping beepers in Fibonacci numbers.

2-1-1. (2 points) Complete the function **turn\_right** which makes the hubo turn right.



```

1 from cs1robots2 import *
2 create_world(avenues=4, streets=4)
3 hubo = Robot (beepers=100000)
4
5 def turn_right():
6     (2-1-1)
7     hubo.turn_left()

```

[Answer box for (2-1-1)]

2-1-2. (6 points) Complete the function **fibonacci\_drop**, which takes a list to keep the Fibonacci count as an argument *count\_list*, drops beepers as much as the next Fibonacci number, and keeps adding the next Fibonacci numbers to the list.

```

1 def fibonacci_drop(count_list):
2     num_beepers = (2-1-2-1)
3     for (2-1-2-2)
4         hubo.drop_beeper()
5
6     (2-1-2-3)

```

[Answer box for (2-1-2-1)]

[Answer box for (2-1-2-2)]

[Answer box for (2-1-2-3)]

2-1-3. (6 points) Let's put everything together. Please complete the program below, so that the hubo moves in the clockwise pattern

```
1 from cs1robots2 import *
2 create_world(avenues=4, streets=4)
3 hubo = Robot(beepers=10000000)
4
5 def turn_right():
6     #your implementation above
7
8 def fibo_drop(count_list):
9     #your implementation above
10
11 def step_back():
12     hubo.turn_left()
13     hubo.turn_left()
14     if hubo.front_is_clear():
15         hubo.move()
16     turn_right()
17     if hubo.front_is_clear():
18         hubo.move()
19
20 count = [0,1]
21
22 while not hubo.on_beeper():
23     if (2-1-3-1)
24         hubo.turn_left()
25
26     (2-1-3-2)
27
28     if not hubo.front_is_clear():
29         break
30     hubo.move()
31
32     if (2-1-3-3)
33         step_back()
```

**[Answer box for (2-1-3-1)]**

**[Answer box for (2-1-3-2)]**

**[Answer box for (2-1-3-3)]**

- 2-2. We want to build a simple library management system. For each book, we decide to maintain a tuple of the book title, location, rental state, return date, and the student ID of the borrower for each book.
- 2-2-1. (4 points) Your friendly TA insisted you need the following function just in case you want to change information about books. Why do you need this function and how does this function work to do the job? Describe in **detail**. [Hint. what is the difference between a tuple and a list?]

```
1 def modify_tuple(tup, index, value):  
2     list_tup = list(tup)  
3     list_tup[index] = value  
4     return tuple(list_tup)
```

[Answer box]



2-2-2. (4 points) The books are stored in a list called *library* as shown below. Complete the function **is\_valid\_book**, which checks if the book exists in the library by checking if the title of the book requested exists in the list *library*.

[Hint. `str.upper()` returns the string with all the alphabetic characters in uppercase. (ex, `"cs101".upper()` → `"CS101"`)]

```
1 library = [ ('Joy of Programming', '1-A', False, None, None),
2             ('The Life of Hubo', '1-B', False, None, None),
3             ('Beauty and the List', '1-C', False, None, None),
4             ('Your (Variable) Name', '1-D', False, None, None),
5             ('Elice in Wonderland', '2-A', False, None, None),
6             ('Frozen: the infinite loop', '2-C', False, None,
7             None),
8             ('Avatar: pair programming', '2-D', False, None,
9             None)]
10
11 def is_valid_book(library, title):
12     if title is None:
13         return False
14
15     book_titles = (2-2-2-1)
16
17     for book_title in book_titles:
18         (2-2-2-2)
19
20     return False
```

[Answer box for (2-2-2-1)]

[Answer box for (2-2-2-2)]

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

3-1. (6 points) The following program prints out multiplication tables. You are asked to rewrite the following program using while-loop instead of for-loop and if statements. Your program must print out the same output as the existing program. If you use the keyword, 'if' or 'for' in your program, you will get 0 point.

cf) The **end=" "** command at the rightmost of the print( ) statement is an option to print without a new line.

**[Program]**

```
1 for i in range(10) :
2     if i%2==1:
3         for j in range(10) :
4             if j%2==1:
5                 print ( " ", i, "*", j, "=", i*j, end=" ")
6             print ("\n")
```

**[Output]**

```
1  1 * 1 = 1   1 * 3 = 3   1 * 5 = 5   1 * 7 = 7   1 * 9 = 9
2
3  3 * 1 = 3   3 * 3 = 9   3 * 5 = 15   3 * 7 = 21   3 * 9 = 27
4
5  5 * 1 = 5   5 * 3 = 15   5 * 5 = 25   5 * 7 = 35   5 * 9 = 45
6
7  7 * 1 = 7   7 * 3 = 21   7 * 5 = 35   7 * 7 = 49   7 * 9 = 63
8
9  9 * 1 = 9   9 * 3 = 27   9 * 5 = 45   9 * 7 = 63   9 * 9 = 81
```

**[Answer box]**

3-2. (14 points) You have to write code to make the robot object move in the zigzag fashion. The below program assumes 5x5 empty world, but your program code should work at any size of empty world. Answer the following questions.

**[Program]**

```

1 from cslrobots import *
2 create_world(5,5) # It should work in an empty world of any size!!!
3 hubo = Robot()
4 hubo.set_trace('blue')
5
6 def turn_right():
7     for i in range(3):
8         hubo.turn_left()
9
10 def march():
11     while hubo.front_is_clear():
12         hubo.move()
13
14 def facing_south():
15     (3-2-1):
16
17 while hubo.front_is_clear():
18     if hubo.facing_north():
19         march()
20         (3-2-2):
21     elif facing_south():
22         march()
23         (3-2-3):
24     else:
25         (3-2-4):

```

Table 1: Member functions of the Robot class related to (3-2)

Function	Description
<b>front_is_clear()</b>	Returns True if no wall in front of robot, False otherwise.
<b>left_is_clear()</b>	Returns True if no walls are to the immediate left of the robot, False otherwise.
<b>right_is_clear()</b>	Returns True if no walls are to the immediate right of the robot, False otherwise.
<b>facing_north()</b>	Returns True if Robot is facing north, False otherwise.

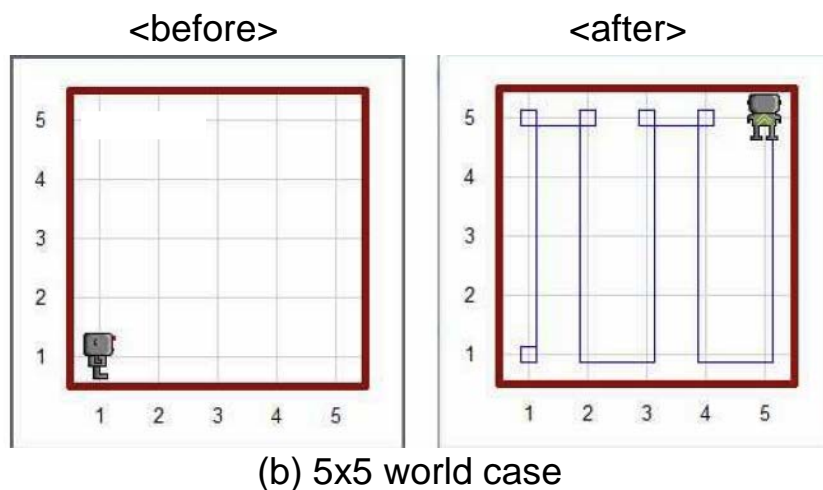
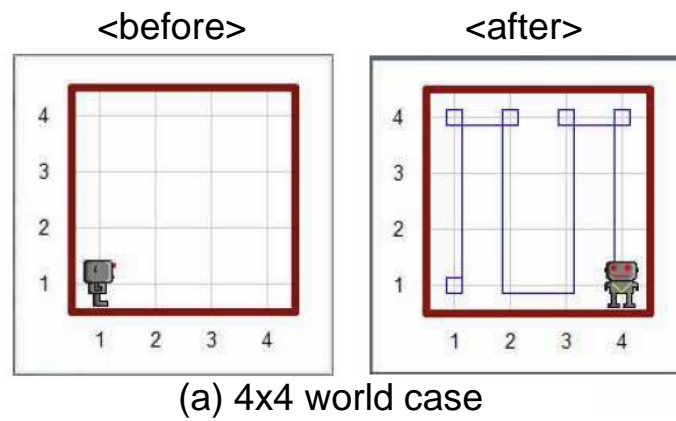


Figure 1: Example results of 'Zigzag' task.

3-2-1. (6 points) Write the function 'facing\_south()' that which allows the robot to tell if he is facing in the south direction or not. Hint) use a built-in function facing\_north().

**[Answer box]**

3-2-2. (3 points) Fill in the blanks to produce the same result as output.

**[Answer box]**

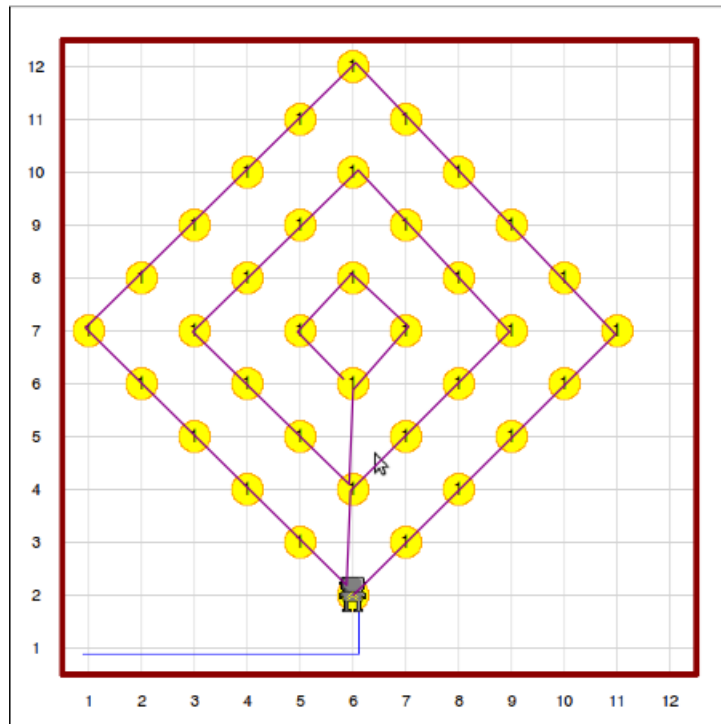
3-2-3. (3 points) Fill in the blanks to produce the same result as output.

**[Answer box]**

3-2-4. (2 points) Fill in the blanks to produce the same result as output.

**[Answer box]**

4. (20 points) Answer each question according to the instruction.
- 4-1. (7 points) We learned about the function. We also practically use that to improve the programming logic following a harvesting problem (4-1a).



**[4-1a]**

**[4-1b]**

```

1 def stairs(robot, n):
2     for i in range(n):
3         robot.pick_beeper()
4         robot.move()
5         turn_right(robot)
6         robot.move()
7         robot.turn_left()
8
9 def diamond(robot, n):
10    for i in range(4):
11        stairs(robot, n)
12        robot.turn_left()
13
14 def harvest_all(robot):
15    for i in range(3):
16        n=5-2*i
17        diamond(robot, n)
18        robot.move()
19        robot.move()

```

```

1 def stairs(robot, n):
2     for i in range(n):
3         robot.pick_beeper()
4         robot.move()
5         turn_right(robot)
6         robot.move()
7         robot.turn_left()
8         (4-1-1):
9
10 def diamond(robot, n):
11     while (4-1-2):
12         robot.turn_left()
13
14 def harvest_all(robot):
15     for i in range(3):
16         n=5-2*i
17         diamond(robot, n)
18         robot.move()
19         robot.move()

```

We want to change the for-loop iteration of the *diamond* function to the while-loop. Rewrite the following code to change 4-1a to 4-1b.

**[Answer box for (4-1-1)]**

**[Answer box for (4-1-2)]**

- 4-2. (7 points) The *math* module have a sine function that take an radian input as the parameter. Define a new *sin* function which take a degree input as the parameter and answer the output of the following program.

**[Program]****[Output]**

```
1 import math
2 sin = math.sin
3 a = 1
4
5 print(sin(30))
6 print('hello world')
7 print(sin)
8
9 
10
11 print(sin(30))
```

```
1 -0.9880316240928618
2 hello world
3 
4 0.5
```

**[Answer box for (4-2-1)]****[Answer box for (4-2-2)]**



- 4-3. (6 points) We learned that an object has state and can perform actions. A function is an object and it is possible to use a function as an argument. Following program shows an example of the feature.

```
1 def f(x):
2     return math.sin(x / 3.0 + math.pi/4.0)
3
4 def print_table(func, x0, x1, step):
5     x = x0
6     while x <= x1:
7         print (x, func(x))
8         x += step
9
10 print_table(f, -math.pi, 3 * math.pi, math.pi/8)
```

Answer the result of the following code with reference to the above.

```
1 def quadratic(a, b, c):
2     def myResult(x):
3         quad_term = a * x ** 2
4         lin_term = b * x
5         return quad_term + lin_term + c
6     return myResult
7
8 myfunc1 = quadratic(1,4,5)
9 myfunc2 = quadratic(2,4,2)
10 print(myfunc1(2))
11 print(myfunc2(2))
```

[Answer box]

5. (20 points) Answer each of the following questions according to the instruction.

5-1. (6 points) The following program contains some faults. Find the line numbers of the faults, and then fix them so that the program shows the example behaviour in the 'output box' below. Note that you are not allowed to 'hard-code' the example behaviour: when different input values are used, the corrected program should output different results.

**[Program]**

```
1 import math
2
3 def myTest1 (radian):
4     sin = math.sin
5     print ("- sin(" + str(radian) + ") is " + str( sin( radian ) ) )
6
7 def myTest2 (degree):
8     pi = math.pi
9     radian = 2 * pi * degree / 360
10    print ( "- Degree " + str(degree) + " is " + str(radian) + "
    radian")
11
12 def myTest3 (div):
13    print ( "- cos(pi/" + str(div) + ") is " + str(math.cos(pi/div))
    )
14    print ( "- sin(pi/" + str(div) + ") is " + str(sin(pi/div)) )
15
16 val1 = int ( input ( "Enter a number1 : " ) )
17 myTest1 ( pi/val1 )
18
19 val2 = int ( input ( "Enter a number2 : " ) )
20 myTest2 ( val2 )
21
22 val3 = int ( input ( "Enter a number3 : " ) )
23 myTest3 ( val3 )
```

**[Output box]**

```
Enter a number1 : 6
- sin(0.5235987755982988) is 0.49999999999999994
Enter a number2 : 30
- Degree 30 is 0.5235987755982988 radian
Enter a number3 : 4
- cos(pi/4) is 0.7071067811865476
- sin(pi/4) is 0.7071067811865475
```

**[Answer box]**

Lines :

5-2. (6 points) What is the output of the following program?

**[Program]**

```
1 def double():
2     global cnt
3     cnt *= 2
4
5 def reset(n):
6     cnt = n
7
8 def test (upper):
9     while cnt < upper:
10        double()
11        if cnt > upper:
12            reset(upper)
13        return
14
15 cnt = 1
16 for i in range(3):
17     double()
18 print ("cnt is " + str(cnt))
19
20 reset(7)
21 double()
22 print ("cnt is " + str(cnt))
23
24 cnt = 5
25 test(20)
26 print ("cnt is " + str(cnt))
```

**[Answer box]**

- 5-3. (8 points) The following program will draw some objects in the canvas. At the end of its execution, which color will be shown at each position specified below? (There is an empty canvas on the next page for you to practice.)

**[Program]**

```
1 from cslgraphics import *
2
3 paper = Canvas(300, 200, 'skyblue')
4
5 sq = Square(50, Point(75, 75))
6 sq.setBorderColor("black")
7 sq.setDepth(40)
8 paper.add(sq)
9
10 po = Polygon(Point(150, 25), Point(225, 25), Point(275, 200), Point
    (150, 150))
11 po.move(-25, 0)
12 po.setBorderWidth(2)
13 po.setBorderColor("green")
14 po.setFillColor("red")
15 po.setDepth(30)
16 paper.add(po)
17
18 cir = Circle(40)
19 cir.setFillColor("gray")
20 cir.setBorderColor("pink")
21 cir.setDepth(60)
22 cir.move(175, 75)
23 paper.add(cir)
24
25 rec = Rectangle(20, 200, Point(150, 25))
26 rec.rotate(90)
27 rec.setFillColor("purple")
28 rec.setBorderColor("yellow")
29 rec.setDepth(20)
30 paper.add(rec)
```

**[Answer]**

Point (75, 75) : \_\_\_\_\_

Point (150, 150) : \_\_\_\_\_

Point (175, 75) : \_\_\_\_\_

Point (225, 25) : \_\_\_\_\_

