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

[CS101] Introduction to Programming 2014 Spring – Final Examination

SECTION	STUDENT ID	NAME

※ Please check to see if you have all **18 pages** in your test material.
※ 시작하기 전에 반드시 페이지의 수를 확인 하십시오. (전체 : **18 쪽**)
※ Fill in your section, student identification number and name. Or you will lose 1 point for each missing piece of information
※ 위의 정보(분반,학번,이름)를 정확히 기입하지 않을 경우, 각 실수 당 1점이 감점 됩니다.
※ **We will not answer questions about the problems during this exam**. If you think there is anything ambiguous, unclear or wrong about a problem, please write down your reasons and make necessary assumptions to proceed with the problem. We will take your explanation into consideration when 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.

1. (20 points) Answer each question according to the instruction.

1-1. (5 points) What is the results of the following program.

(※ Each result should be matched with given sub-number marked as comment.)

(※ Your answer should be filled into the given table in a proper way as the answer for '1-1-0' is written in the given table.)

```
a = ' duck duck'
print a                                    # 1-1-0 – example
print a.strip()                            # 1-1-1
print len(a[-2:])                          # 1-1-2
print a.find('DUCK') + len(a.split())      # 1-1-3
print '5'.join(a.split(' '))               # 1-1-4
print 'u' in list(a)                       # 1-1-5
```

1-1-0		d	u	c	k		d	u	c	k		
1-1-1 (1 point)												
1-1-2 (1 point)												
1-1-3 (1 point)												
1-1-4 (1 point)												
1-1-5 (1 point)												

1-2. (4 points) What is the result of the following program?

```
class Champion():
    pass
nami = Champion()
teemo = nami
good_team = False

def func1(p1, p2):
    good_team = p1 is p2
    print good_team

func1(nami, teemo)
print good_team
```

____

____

1-3. (5 points) A function *'myCount'* is defined to count the number of repetition of the string parameter, *pattern*, in the string parameter, *stream*, which performs exactly same as *str.count()* function. **Select all correct codes** for the blank which produces the result as shown below. Mark 'O' for the correct answer, leave it empty blank otherwise. (**Multiple choices, No partial score**)

```
def myCount(stream, pattern):
    counter = 0
    
                    #1-3
    
    return counter
bit_stream = '0110011101001'
print myCount(bit_stream,'01')
```

Result:

4

Given code	Your answer
`counter = stream.count(pattern)`	O

`while len(stream) >0:`
`    if pattern in stream:`
`        counter += 1`
`        stream = stream - pattern`	

`while pattern in stream:`
`    stream = stream[stream.find(pattern) + len(pattern):]`
`    counter += 1`	

`for i in range(len(stream) - len(pattern)+1):`
`    if stream[i : i+len(pattern)] == pattern:`
`        counter += 1`	

`for word in stream:`
`    if word == pattern:`
`        counter += 1`	

1-4. (3 points) What is the result of the following program?

```
k = [1,2,3,4,5]
t = 3
idx = 0
while not k[idx] == t:
    print k[idx],
    idx = idx - 1
```

____

1-5. (3 points) What is the result of the following program?

```
for a in range(2):
    for b in range(2):
        if a==b:
            continue
        print 1,
    print 2,
print 3
```

____

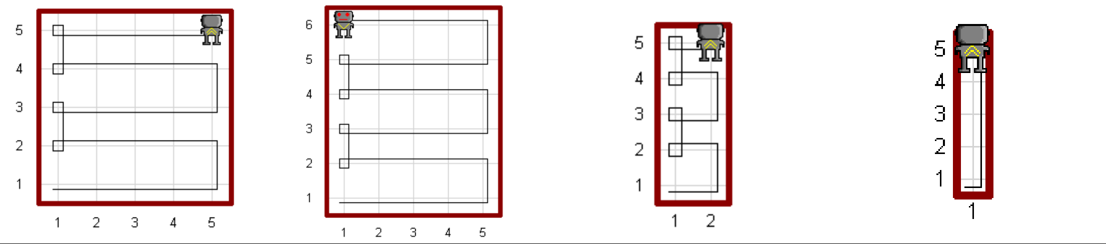
2. (20 points) Answer each question.

2-1. (6 points) Fill in the blank so that the 'robot' can visit every place in an empty world of any size. Notice that, in the following program, the 'robot' should visit every place only once.

Function name	**Description** (defined in ***'random'*** module)
randrange	Usage: random.randrange([start], stop[, step]) Return a randomly selected element from range(start, stop, step).

example	>>> from random import randrange
	>>> randrange(1, 20)
	12
	>>> randrange(1, 20)
	5
	>>> randrange(1, 20)
	1

Function name	**Description** (defined for ***'Robot'*** object in ***'cs1robots'*** module)
carries_beepers	Returns ***True*** if some beepers are left in Robot's bag.
drop_beeper	Robot drops one beeper down at current location.
facing_north	Returns ***True*** if Robot is facing north.
front_is_clear	Returns ***True*** if no wall or border in front of robot.
left_is_clear	Returns ***True*** if no walls or borders are to the immediate left of the robot.
move	Move one street/avenue in direction where robot is facing.
on_beeper	Returns ***True*** if beepers are present at current robot position.
pick_beeper	Robot picks one beeper up at current location.
right_is_clear	Returns ***True*** if no walls or borders are to the immediate right of the robot.
turn_left	Rotate left by 90 degrees.

Example


```
from cs1robots import *
from random import randrange

create_world(streets = randrange(1, 20), avenues = randrange(1, 20))

hubo = Robot()
hubo.set_trace('black')

def turn_right(robot):
    for i in range(3):
        robot.turn_left()

def go_straight(robot):
    while robot.front_is_clear():
        robot.move()

def zigzag(robot):
    while True:
        if robot.front_is_clear():
            go_straight(robot)
            if robot.left_is_clear():
                robot.turn_left()
                robot.move()
                robot.turn_left()
                go_straight(robot)
                if robot.right_is_clear():
                    turn_right(robot)
                    robot.move()
                    turn_right(robot)
        else:
                                    #2-1
            return

zigzag(hubo)
```

2-2. (6 points) After execution of the following program, draw positions of 'hubo' and 'ami' with their directions where they are looking at, respectively. Notice that the position of 'hubo' (a blue Robot object) should be □ and the position of 'ami' (a yellow Robot object) should be ○ with directions ↑,↓,←, or →, as the following example.

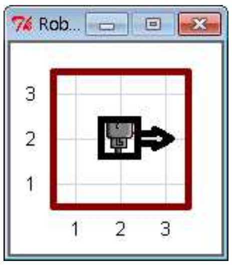
Example

```
from cs1robots import *

create_world(streets = 3, avenues = 3)

hubo = Robot(street = 2, avenue = 2)
```

$\Rightarrow$

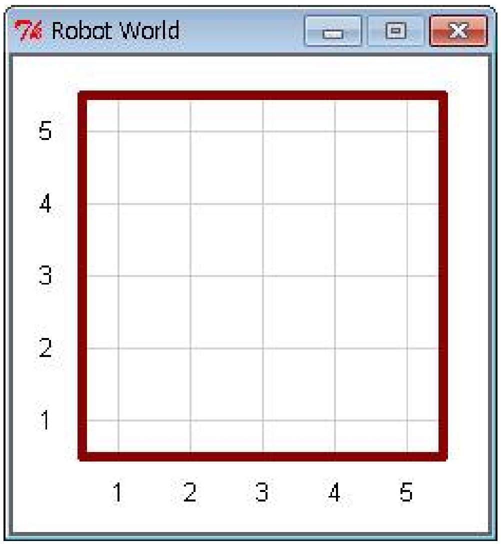


```
from cs1robots import *

create_world(streets = 5, avenues = 5)

hubo = Robot(street = 3, avenue = 3, color = 'yellow')
hubo.move()
ami = hubo
ami.turn_left()
hubo.move()

hubo = Robot(street = 3, avenue = 3, color = 'blue')
hubo.move()
ami.turn_left()
ami.move()
```



2-3. (6 points) A function can only return one value. But functions can return arbitrarily many values. Write down the result of the following program.

```
def swap(a, b):
    b, a = a, b
    return a, b

w = 12
x = 34
y = 56
z = 78

result = swap(swap(w, x), swap(y, z))

print type(swap)
print type(result)
print result
```

<type ‘function’>

____

____

2-4. (2 points) Write down the result of the following program.

```
nobles = ['helium', 'noen', 'krypton', 'xenon']

fixed_nobles = nobles
fixed_nobles[1] = 'neon'

appended_nobles = fixed_nobles[:]

print nobles is appended_nobles
```

____

3. (20 points) player.txt and map.txt contain information of players and terrains, respectively. In player.txt, each line contains information of a player. The player's name, age, running skill, swimming skill, and climbing skill are separated with a comma. map.txt shows terrains: 1 = plain, 2 = water, and 3 = mountain. Players cannot move through a time 0 which indicate a steep cliff.

[Files]

player.txt	map.txt
John,25,1,2,3 Smith,22,2,1,3 James,29,3,2,1	**201111330** **201000020** **201000020** **201000020** **201001120** **201003000** **222003110**

[Program]

```
class Player(object):
    name = "" # the name of this player, string
    age = -1  # the age of this player, int
    run = -1  # the running skill of this player, int
    swim = -1 # the swimming skill of this player, int
    climb = -1 # the climbing skill of this player, int
    index = -1 # the index of this player on adv_map, int
    pos = None # the position of this player, (x,y)
    day = -1   # the number of elapsed days of this player, int

    def __init__(self,name,age,run,swim,climb):
        #Assume that already implemented __init__ correctly
    def __cmp__(self,other):
        #Assume that already implemented __cmp__ correctly
    def move(self):
        #Assume that already implementd move correctly
    def select_equip(self):
        [                    #3-2                    ]

def make_map():
    [                  #3-1                  ]

adv_map = make_map(m_filename)
```

3-1. (10 points) Function *make_map(m_filename)* reads file whose name is *m_filename* and make a list of tuples indicating a journey path. Each tuple consists of (x,y) and z, x = x coordinate value, y = y coordinate value and z = terrain value. Combining the available candidates, implement *make_map* function to work correctly.
(e.g.) A/B/C, A/2/C/4, C/4, etc...

[Expected result of *make_map("map.txt")*]

```
[((0, 0), 2), ((0, 1), 2), ((0, 2), 2), ((0, 3), 2), ((0, 4), 2), ((0, 5), 2), ((0,
6), 2), ((1, 6), 2), ((2, 6), 2), ((2, 5), 1), ((2, 4), 1), ((2, 3), 1), ((2, 2),
1), ((2, 1), 1), ((2, 0), 1), ((3, 0), 1), ((4, 0), 1), ((5, 0), 1), ((6, 0), 3),
((7, 0), 3), ((7, 1), 2), ((7, 2), 2), ((7, 3), 2), ((7, 4), 2), ((6, 4), 1), ((5,
4), 1), ((5, 5), 3), ((5, 6), 3), ((6, 6), 1), ((7, 6), 1)]
```

[Available candidates]

(A)	`if map_list[y][x+1] and (not x+1 == priv_x or not y == priv_y):`
	`    ret_list.append(((x+1,y),map_list[y][x+1]))`
	`    priv_x = x`
	`    priv_y = y`
	`    x += 1`
(B)	`if map_list[y][x-1] and (not x-1 == priv_x or not y == priv_y):`
	`    ret_list.append(((x-1,y),map_list[y][x-1]))`
	`    priv_x = x`
	`    priv_y = y`
	`    x -= 1`
(C)	`if map_list[y+1][x] and (not x == priv_x or not y+1 == priv_y):`
	`    ret_list.append(((x,y+1),map_list[y+1][x]))`
	`    priv_x = x`
	`    priv_y = y`
	`    y += 1`
(D)	`if map_list[y-1][x] and (not x == priv_x or not y-1 == priv_y):`
	`    ret_list.append(((x,y-1),map_list[y-1][x]))`
	`    priv_x = x`
	`    priv_y = y`
	`    y -= 1`
(1)	`elif map_list[y][x+1] and (not x+1 == priv_x or not y == priv_y):`
	`    #Remaining code is same with (A)`
(2)	`elif map_list[y][x-1] and (not x-1 == priv_x or not y == priv_y):`
	`    #Remaining code is same with (B)`
(3)	`elif map_list[y+1][x] and (not x == priv_x or not y+1 == priv_y):`
	`    #Remaining code is same with (C)`
(4)	`elif map_list[y-1][x] and (not x == priv_x or not y-1 == priv_y):`
	`    #Remaining code is same with (D)`

```
def make_map(m_filename):
    map_list = []
    ret_list = []
    f = open(m_filename)
    for line in f:
    |   line = line.strip()
    |   temp_list = []
    |   for i in line:
    |   |   temp_list.append(int(i))
    |   map_list.append(temp_list)
    x = 0
    y = 0
    priv_x = -1
    priv_y = -1
    s_tuple= ((x,y),map_list[0][0])
    ret_list.append(s_tuple)
    state = True
    while state:
    |   if x == 0:
    |   |  if y == 0:
    |   |  |   #3-1-1
    |   |  |    else:
    |   |  |    |   state = False
    |   |  elif y == len(map_list)-1:
    |   |  |   #3-1-2
    |   |  |   else:
    |   |  |   |    state = False
    |   |  else:
    |   |  |   #3-1-3
```

```
    |   |  |    else:
    |   |  |    |   state = False
    |   elif x == len(map_list[0])-1:
    |   |  if y == 0:
    |   |  |    #3-1-4
    |   |  |   else:
    |   |  |   |    state = False
    |   |  elif y == len(map_list)-1:
    |   |  |    #3-1-5
    |   |  |   else:
    |   |  |   |    state = False
    |   |  else:
    |   |  |    #3-1-6
    |   |  |    else:
    |   |  |    |   state = False
    |   else:
    |   |  if y == 0:
    |   |  |    #3-1-7
    |   |  |    else:
    |   |  |    |   state = False
    |   |  elif y == len(map_list)-1:
    |   |  |    #3-1-8
    |   |  |    else:
    |   |  |    |   state = False
    |   |  else:
    |   |  |    #3-1-9
    |   |  |    else:
    |   |  |    |   state = False
    return ret_list
```

3-1-1	
3-1-2	
3-1-3	
3-1-4	
3-1-5	
3-1-6	
3-1-7	
3-1-8	
3-1-9	

3-2. (10 points) Method *select_equip(self)* of *Player* class return the most useful equipment number for current player and increases the relevant skill of the player by 1 by using *adv_map* variable created by *make_map* function. For example, current player's running is 2, swimming is 1, and climbing skill is 1. He selects swimming equipment at the bold-underscored line, because 3/2+6/(1+1)+3/1 is smaller than 3/(2+1)+6/1+3/1, and 3/2+6/1+3/(1+1). The *select_equip* method will increase 1 of *swim* attribute of current player and return 2.

Implement *select_equip* method.

1	1	1	1	1	**1**	**2**	**2**	**2**	**3**	**3**	**3**	**1**	**1**	**2**	**2**	**2**

Example of terrain info of *adv_map* variable (→ Remaining tiles)

```
def select_equip(self):
    # output: 1 if player selects running equipment
    #         2 if player selects swimming equipment
    #         3 if player selects climbing equipment
```

4. (20 points) Answer each question according to the instruction.

4-1. (2 points) What is the result of the following program?

```
result = 0
for outer in [1, 2, 3, 4, 5] :
    if outer == 3 :
        continue
    for inner in [1, 2, 3] :
        if outer == 2 :
            break
        result += inner
print result
```

____

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

```
tokens = "CS,101".split(",")
print tokens[0].join(tokens)
```

____

4-3. (4 points) The function 'cat' merges a set of input files into a single output file. (That is, all the contents in a set of the input files are put into the single output file.) For the function 'cat', names of a set of input files are given in a sequence **(list of strings)** as the first argument and a name of a single output file (**string**) is given as the second argument.

Complete the following function 'cat'.

```
def cat(input_files, output_file) :













# an example of the function call
cat(["a.txt", "b.txt", "c.txt"], "result.txt")
```

4-4. (6 points) The following program counts the number of palindromic words in a text file. A palindromic word is a word whose meaning may be interpreted in the same way in both forward and reverse direction. Some examples of common palindromic words are 'civic', 'level', 'noon' and 'maddam'.

For simplicity, we assume that each line of the text file represents a single word. In other words, there can only be line separator('\n') between words.

Fill in the blanks.

```
# input : a string
# output : a boolean determined according to whether or not the given word
# is palindromic

def is_palindromic(word) :
    for i in range(len(word)/2) :
        if [          #4-4-1          ] :
            return False
    return True

count = 0
fin = open(raw_input("file name -> "), "r")

for [          #4-4-2          ] :
    if [          #4-4-3          ] :
        count += 1
fin.close()

print count
```

4-4-1	
4-4-2	
4-4-3	

4-5. (6 points) ‘countries.csv’ contains country code, country name and latitude which are comma-separated as follows.

[File]

```
countries.csv
Code,Name,Latitude
AU,Australia,-27
KR,Korea, Repulic of,37
US,United States,38
BR,Brazil,-10
```

Fill in the blanks so that the following function, 'load_countries' returns a list of tuples of strings as shown in the result box. The last element of a tuple means the country's location based on the equator.

[Program]

```
# input : a string
# output : a list of tuples of strings

def load_countries(filename) :
    ctr = []
    fin = open(filename, "r")
    fin.readline()
    for line in fin :
        t = [          #4-5-1          ]
        if [          #4-5-2          ] :
            t.append("North")
        else :
            t.append("South")
        if [          #4-5-3          ] :
            ctr.append((t[0], t[1] + "," + t[2], t[-2], t[-1]))
        else :
            ctr.append((t[0], t[1], t[-2], t[-1]))
    fin.close()
    return ctr

countries = load_countries("countries.csv")
```

[Result]

countries[0]	('AU', 'Australia', '-27', 'South')
countries[1]	('KR', 'Korea, Repulic of', '37', 'North')
countries[2]	('US', 'United States', '38', 'North')
countries[3]	('BR', 'Brazil', '-10', 'South')

4-5-1	
4-5-2	
4-5-3	

5. (20 points) In this problem you are asked to implement Circle and Ring class. Circle class is defined as its radius and Ring class is defined as two circle object.

Please fill in blanks to complete the class.

```
class Circle(object):
        [            #5-1            ]


        [            #5-2            ]


class Ring(object):
        [            #5-3            ]


        [            #5-4            ]
```

5-1. (3 points) Fill in the blank with the constructor method of *Circle* class. Constructor has a parameter: *radius* (Integer value). You must truncate radius value. That is, it must be in range from 1 to 100. If the radius is smaller than 1, it must be assigned as 1. But, if the radius is larger than 100, it must be assigned as 100.

5-2. (6 points) Fill in the blank to write two functions of Circle; *getRadius()* and *getArea()*. Its usage is below. You can assume that math module is already imported.

```
>>> c1 = Circle(2)
>>> print c1.getRadius()
2
>>> print c1.getArea()
12.5663706144
```

```
import math
```

5-3. (6 points) Fill in the blank with the constructor method of *Ring* class. Constructor has two parameters: inner_circle and outer_circle (which are Circle objects). You must check that the outer_circle's radius is larger than the inner_circle's. If not, swap the inner_circle and the outer_circle. You must use *getRadius()* function of *Circle* class in the constructor. (You don't need to consider exactly same radius of two circles)

5-4. (5 points) Fill in the blank to write a function of *Ring* class; *getArea()*. Ring's area is measured by outer_circle's area minus inner_circle's area. You must use *getArea()* function of *Circle* class in this function. An example of the use of *getArea()* is given below.

```
>>> c1 = Circle(2)
>>> c2 = Circle(5)
>>> r = Ring(c1,c2)
>>> print r.getArea()
65.9734457254
```