**[OFFICIAL USE ONLY]**

| 1 (21) | 2 (18) | 3 (19) | 4 (20) | 5 (22) | TOTAL (100) |
|---|---|---|---|---|---|
| | | | | | |

# CS101 Introduction to Programming 2011 Fall Final Examination

| SECTION | STUDENT ID | NAME |
|---|---|---|
| | | |

※ Please check to see if you have all 16 pages in your test material.
※ 시작하기 전에 반드시 페이지의 수를 확인 하십시오. (전체 : 16쪽)
※ Fill in your student identification number and name. Or you will lose 1 point for each missing piece of information.
※ 학번 및 이름을 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. Answer each of the following questions according to the instruction.

**(21 points)**

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

```
>>> print type(("1",456)) # (1 point)
                                                  <type '_________'>
>>> print type("123 456 789".split()) # (1 point)
                                                  <type '_________'>
>>> l = [2,3,4,"Ace","Queen"]

>>> print type(l[3]) # (1 point)
                                                  <type '_________'>
>>> print len(l) # (1 point)
                                                  __________________
>>> print len(l[:-1]) # (1 point)
                                                  __________________
```

1-2. Fill out the box with "T" if the answer is true and with "F" for false. You also can choose "IDK" if you don't know the answer. If your answer is correct, then you will get 1 point. Otherwise, you will lose 2 points. "IDK" or a blank will not be counted as any point.

(5 points)

| Statement | Answer |
|---|---|
| (Example) Variables defined outside of a function are called ***global variables***. | T |
| ***User-defined objects*** are a kind of sequence. | |
| ***The cmp(a,b) function*** which is used in *sort()* returns True or False. | |
| List and string are very similar, but ***only string is mutable.*** | |
| The keyword 'continue' ***only used in for-loop statement.*** | |
| Prof. YoonJoon Lee is ***handsome***. | |

1-3. What is the result of the following program?

(3 points)

```
s = [1,2,3]
t = ['a',s,'c']
print s # (1 point)
print t[1][2] # (1 point)
s[1] = 200
print t # (1 point)
```

1-4. What is the result of the following program?

(5 points)

```
f = "I don't like CS101"
items = f.split()
print items # (1 point)
print items.pop(1) # (1 point)

message = ' '.join(items)
print message # (1 point)
print type(message) # (1 point)

f[:-1]
print f # (1 point)
```

1-5. How many attributes does the instance of Card object have after executing the last statement?

(3 points)

```
class Card(object):
   def __init__(self):
      self.name="Ace"
      self.num = 11
   def makeAttr(self):
      self.score = 100

card = Card()
card.state = "True"
```

**2. Define a class so that the following codes make the following output.**

**(18 points)**

[Program Code]

```
a = TA("JH", 30, "M")
b = TA("YJ", 29, "M")
c = TA("JY", 26, “W”)
d = TA("YJ", 29)
e = TA("YJ", 27)
print a == b
print b == d
print b == e
print b != d
print a
print b
print c
```

[Output]

```
False
True
False
False
I am JH. I'm 30 years old. I'm man.
I am YJ. I'm 29 years old. I'm man.
I am JY. I'm 26 years old. I'm woman.
```

```
class 
   def __init__ (                                        ):



   def              (self, rhs):



   def              (self, rhs):
      return not (self == rhs)

   def                          :
```

**3. The below file *iphone_app.txt* includes some information about four different iPhone Apps. Please complete the following program to access the file and manipulate its content according to the following instructions.**

**(19 points)**

[iphone_app.txt]

```
#### iPhone Apps Info ####

Infinity Blade Ⅱ($10.99)
Rank :
Today 1       Yesterday 201
Customer Ratings:
Average rating  4.5   ( 13175 Ratings )
Description:
The award-winning Infinity Blade story continues.

Angry Birds($1.99)
Rank :
Today 3       Yesterday 2
Customer Ratings:
Average rating  4.5   ( 631058 Ratings )
Description:
The survival of the Angry Birds is at stake.
Dish out revenge on the green pigs who stole the Birds'eggs.

Instagram(Free)
Rank :
Today 25      Yesterday 25
Customer Ratings:
Average rating  5.0   ( 203772 Ratings )
Description:
11 million users love Instagram!
It's a free, fun, and simple way to make and share gorgeous photos on your iPhone.

TurboScan($11.99)
Rank :
Today 16      Yesterday 26
Customer Ratings:
Average rating  4.0   ( 728 Ratings )
Description:
TurboScan turns your iPhone into a multipage scanner for documents, receipts, whiteboards, business cards, etc.
```

[Program Code]

```
apps = []
def extract_info():
    name = None
    price = 0.00
    rank_fluc = 0
    star = 0.0
    ratings = 0

    f = open("iphone_app.txt", "r")
    line = f.readline()
    while line:
        line = f.readline()
        if
                         (3-1)

        elif "Rank" in line:
                         (3-2)

        elif "Customer Ratings" in line:
            line = f.readline()
            rat = line.strip().split()
            star = float(rat[2])
            ratings = int(rat[4])
            apps.append((name, price, rank_fluc, star, ratings))
    f.close()
def write_file():
                         (3-3)

extract_info()
write_file()
```

```
>>> print apps
[('Infinity Blade II', 10.99, '+200', 4.5, 13175), ('Angry Birds', 1.99,
'-1', 4.5, 631058), ('Instagram', 0.0, '0', 5.0, 203772), ('TurboScan',
11.99, '+10', 4.0, 728)]
```

3-1. Fill in the blank to extract the name and the price of each app from the above *iphone_app.txt*. You should reference the above result *print apps*. (You can use either **if**(**elif**) or **while** statement at most once. If you use logical operators either **and** or **or**, you will get some penalty.) (7 points)

```
if
```

3-2. Fill in the blank to extract Today's rank and Yesterday's rank of each app from the above *iphone_app.txt* and to get the ranking fluctuation calculated by subtracting Today's rank from Yesterday's rank. You should reference the above result *print apps*. (You can use **if**(**elif**) or **while** statement at most once.) (6 points)

```
elif "Rank" in line:
```

3-3. Create a new file *output.txt* and write the data in the list *apps* with a tabular format like the following. (You must use the string formatting operator **%**.. Use field width only to print spaces. You do not have to worry about the number of spaces).

(6 points)

```
Infinity Blade II        10.99   +200          4.5        13175
Angry Birds               1.99     -1          4.5       631058
Instagram                 0.00      0          5.0       203772
TurboScan                11.99    +10          4.0          728
```

```
def write_file():
```

**4. Answer each question according to the instruction.**

**(20 points)**

4-1. Write the output of following code

(5 points)

[Program Code]

```
l = []
s = ' honolulu '                # Be aware the space character on both sides
l.append(s[:])
l.append(len(s.lstrip())/len(s))
l.append(s.replace('lu','p',1))
l.append(s.find('p',1)+ len(s.split('lu')))

print "%10s%-5.2f\n%-10s%d" %tuple(l)
```

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

4-2. A palindrome is a word, phrase, number or other sequence of units that can be read the same way in either direction. A function *is_palindrome* takes one string parameter ***s***, and checks whether ***s*** is palindrome or not. If ***s*** is palindrome then return True, else return False. Fill the blank 4-2 to get following result from the program code.

(5 points)

[Program code]

```
def is_palindrome(s):
                         (4-2)
print is_palindrome('appa')
print is_palindrome('!@#$#@!')
print is_palindrome('cs101')
```

[Output]

```
True
True
False
```

```
def is_palindrome(s):
```

4-3. A function *get_pwd* takes one string parameter *pwd*, and returns string of its current working directory, which is the substring between the first colon ':' and '$'. **Please make the program accept any string with the same format. In other words, we do not know the exact location (index number) of ':' and '$' in each string.**. Fill in the blank 4-3 to get following result from the program code.

(5 points)

[Program Code]

```
def get_pwd(pwd):
                        (4-3)
    return ret_str

print get_pwd("zsaver@bonobono.kaist.ac.kr:/usr/local$ ls -al")
print get_pwd("xpenguin@holypsycho.kaist.ac.kr:~/test/23:44$ pwd")
print get_pwd("ijij45@gom.kaist.ac.kr:~/result/3:3/4:4$ uname -a")
```

[Output]

```
/usr/local
~/test/23:44
~/result/3:3/4:4
```

```
def get_pwd(str):

    return ret_str
```

4-4. A function *startswith* takes two string parameters *input_string* and *compare_string*, and returns a boolean value (True or False). If *input_string* starts with *compare_string*, then return True, otherwise return False. Fill in the blank 4-4 to get following output from the program code.

(5 points)

[Program Code]

```
def startswith(input_string,compare_string):
                        (4-4)


print startswith("ABCD","AB")
print startswith("CS101 2011 Fall Final","CS101 ")
print startswith("CS101 2011 Fall Final","CS 101")
```

[Output]

```
True
True
False
```

```
def startswith(input_string,compare_string):
```

**5. Answer each question according to the instruction.**
**※Notice: Your code should not be specific for examples. (Examples are just examples.) (22 points)**

5-1. A function *generate_date_of_birth* is implemented to generate a list of distinct birth dates. Fill in the blank to complete the function following the description and requirements below.

(8 points)

| | Input | Output |
|---|---|---|
| Description | start: start year (int)<br>end: end year (int)<br>num: the number of people (int) | A list of distinct birth dates (int) |

| Requirements |
|---|
| (1) You should check if input parameters meet two conditions like followings :<br>(1-1) 'start' must be less than 'end'<br>(1-2) 'num' must be bigger than zero<br>If (1-1) and (1-2) are not met, empty list should be returned.<br><br>(2) You should specify proper input values for 'randint' function (5-1-2 and 5-1-3) in order to generate 8 digit numbers between ('start' + xxxx) and ('end' + xxxx).<br>(e.g.) if start = 2000 and end = 2010, then from 2000XXXX to 2010XXXX<br><br>(3) The elements in the resulting list should be distinct.<br>(4) The length of the resulting list must be same as input parameter 'num'. |

**Example for 'randint' function in 'random' module**

```
import random
res_list = []
for i in range(10) :
    res_list.append( random.randint(1, 5) )
print res_list
>>> [2, 1, 4, 3, 5, 3, 1, 3, 1, 2]
```

**[ Program Code ]**

```
def generate_date_of_birth( start, end, num ) :
    birth_list = []
    if (1000<=start<10000) and (1000<=end<10000) and [   (5-1-1)   ] :
        while True :
            rand_num = random.randint( [ (5-1-2) ] , [ (5-1-3) ] )
            if [            (5-1-4)            ] :
                birth_list.append( rand_num )
            if [            (5-1-5)            ] :
                break
    return birth_list

print generate_date_of_birth( 2000, 2010, 5 )
```

**[ Result ]**

```
>>> [20092588, 20031217, 20030229, 20040229, 20070631]
```

5-1-1. (2 points)

5-1-2. (1 point)

5-1-3. (1 point)

5-1-4. (2 points)

5-1-5. (2 points)

5-2. There can be invalid dates in the result of the function 'generate_date_of_birth'. (e.g.) 20092588, 20030229, and 20070631.

It is due that the function does not consider how many months and days for each month exist. (Leap year is not also into the consideration.)

| Month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Days | 31 | 28 or 29 | 31 | 30 | 31 | 30 | 31 | 31 | 30 | 31 | 30 | 31 |

So, let's make a function *num_of_days* to count how many days exist for a certain month of a certain year following the description and requirements below.

(5 points)

| | Input | Output |
|---|---|---|
| Description | year: (int)<br>month: (int) | number of days (int) |

| Requirements |
|---|
| (1) If input parameter 'month' is not between 1 and 12, zero should be returned. |
| (2) Check leap year using 'isleap' function in 'calendar' module. |

**Example for 'isleap' function in 'calendar' module**

```
import calendar
res_list = []
for i in range( 2000, 2005 ) :
    res_list.append( (i, calendar.isleap(i)) )
print res_list
>>> [(2000, True), (2001, False), (2002, False), (2003, False), (2004, True)]
```

**[ Program Code ]**

```
def num_of_days( year, month ) :
    days = 0
    if month in (5-2-1) :
        days = 31
    elif month in (5-2-2) :
        days = 30
    elif month == 2 :
            (5-2-3)
    return days

print num_of_days(2009, 25), num_of_days(2004, 2), num_of_days(2007, 6)
```

**[ Result ]**

```
>>> 0 29 30
```

5-2-1. (1 point)

5-2-2. (1 point)

5-2-3. (3 points)

5-3. Now, we can remove invalid dates using *num_of_days* function. So, let's make a function *filter_invalid_dates* to update a given list to the list without invalid dates following the description and requirements below.

(5 points)

| | Input | Output |
|---|---|---|
| Description | date_list: (list) | None |

| Requirements |
|---|
| (1) Since there is no return value from the function, the input list must be updated within the function. |

**[ Program Code ]**

```
def filter_invalid_dates( date_list ) :
    num_of_invalid_dates = 0
    for i in range( len(date_list) ) :
        idx = ( i - num_of_invalid_dates )

        year = date_list[idx] / (5-3-1)
        month = ( date_list[idx] % (5-3-1) ) / (5-3-2)
        day = date_list[idx] % (5-3-2)

        end_day = num_of_days( year, month )
        if (0 >= day) or (day > end_day) :
                (5-3-3)

people_info = [20092588, 20031217, 20030229, 20040229, 20070631]
filter_invalid_dates( people_info )
print people_info
```

**[ Result ]**

```
>>> [20031217, 20040229]
```

5-3-1. (1 point)

5-3-2. (1 point)

5-3-3. (3 points)

5-4. Finally, valid birth dates are made. Let's calculate ages for each birth date following the description and requirements below.

(4 points)

| | Input | Output |
|---|---|---|
| **Description** | date_list: (list)<br>today: (int) | None |

| Requirements |
|---|
| (1) Each element of the list must be changed from 'int' to 'tuple' consisting of two 'int' elements. (e.g.) 20031217 -> (20031217, 7)<br>(2) Since there is no return value from the function, the input list must be updated within the function.<br>(3) A person becomes 1 year old after an year since the birth date.<br>(e.g.) A person is born on Jan 9 in 2010. He/She is 1 year old from Jan 9 in 2011. (He/She is 0 year old between Jan 9 in 2010 and Jan 8 in 2011.) |

**[ Program Code ]**

```
def calculate_age( date_list, today ) :
    for i in range( len(date_list) ) :
        age = (5-4-1)
        date_list[i] = ( (5-4-2) , (5-4-3) )

people_info = [20031217, 20040229]
calculate_age( people_info, 20110204 )
print people_info
```

**[ Result ]**

```
>>> [(20031217, 7), (20040229, 6)]
```

5-4-1. (2 points)

5-4-2. (1 point)

5-4-3. (1 point)