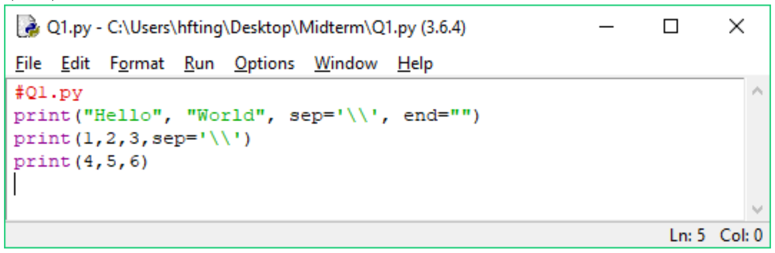
COMP1117A Mid-term

Name: ______________________________________ Student ID: ____________________

What are the outputs of the following programs?

1. (5%)



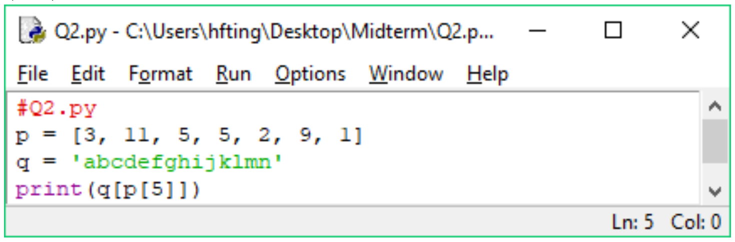
Q1.py - C:\Users\hfting\Desktop\Midterm\Q1.py (3.6.4)

```
#Q1.py
print("Hello", "World", sep='\\', end="")
print(1,2,3,sep='\\')
print(4,5,6)
```

Ln: 5 Col: 0

```
Hello\World1\2\3
4 5 6
```

2. (5%)



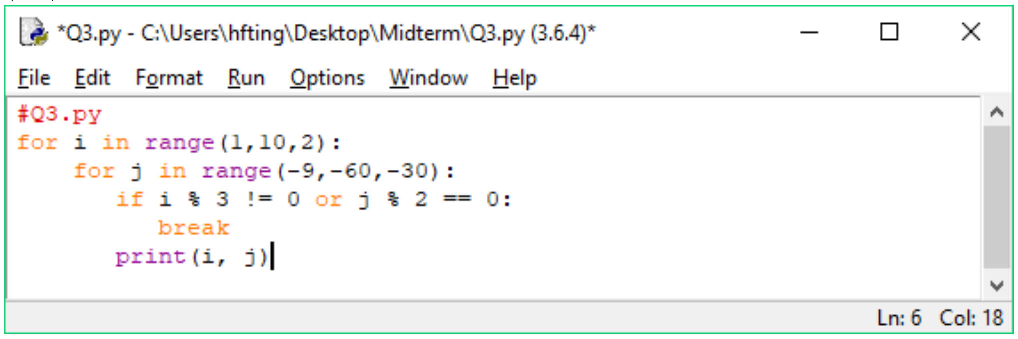
Q2.py - C:\Users\hfting\Desktop\Midterm\Q2.p...

```
#Q2.py
p = [3, 11, 5, 5, 2, 9, 1]
q = 'abcdefghijklmn'
print(q[p[5]])
```

Ln: 5 Col: 0

```
j
```

3. (5%)



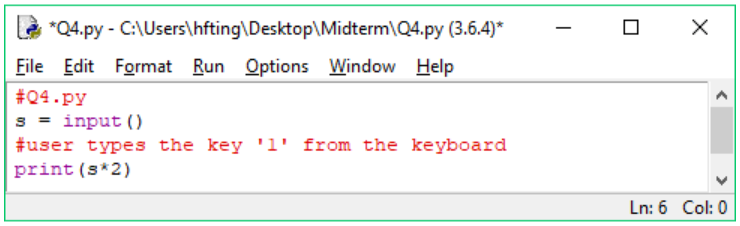
Q3.py - C:\Users\hfting\Desktop\Midterm\Q3.py (3.6.4)

```
#Q3.py
for i in range(1,10,2):
    for j in range(-9,-60,-30):
        if i % 3 != 0 or j % 2 == 0:
            break
        print(i, j)
```

Ln: 6 Col: 18

```
3 -9
3 -39
9 -9
9 -39
```

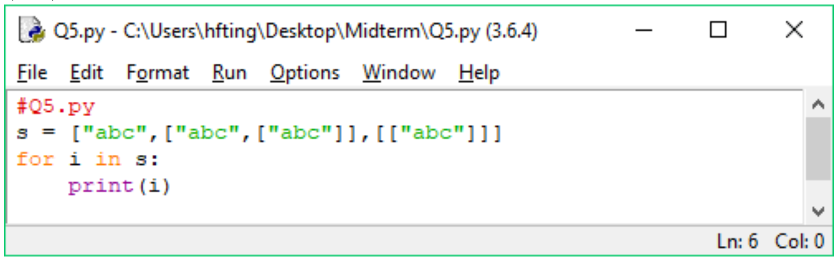
4. (5%)


Q4.py - C:\Users\hfting\Desktop\Midterm\Q4.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 6 Col: 0

```
#Q4.py
s = input()
#user types the key '1' from the keyboard
print(s*2)
```

11

5. (5%)


Q5.py - C:\Users\hfting\Desktop\Midterm\Q5.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 6 Col: 0

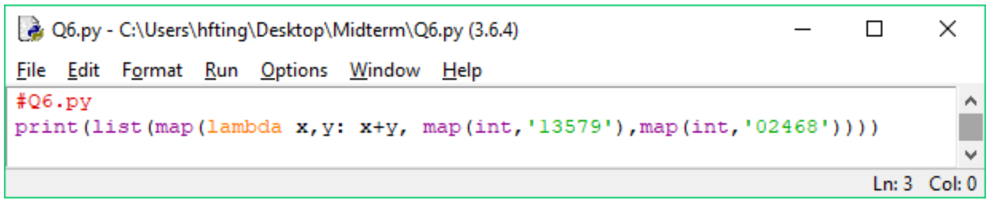
```
#Q5.py
s = ["abc",["abc",["abc"]],[["abc"]]]
for i in s:
    print(i)
```

abc

[‘abc’, [‘abc’]]

[[‘abc’]]

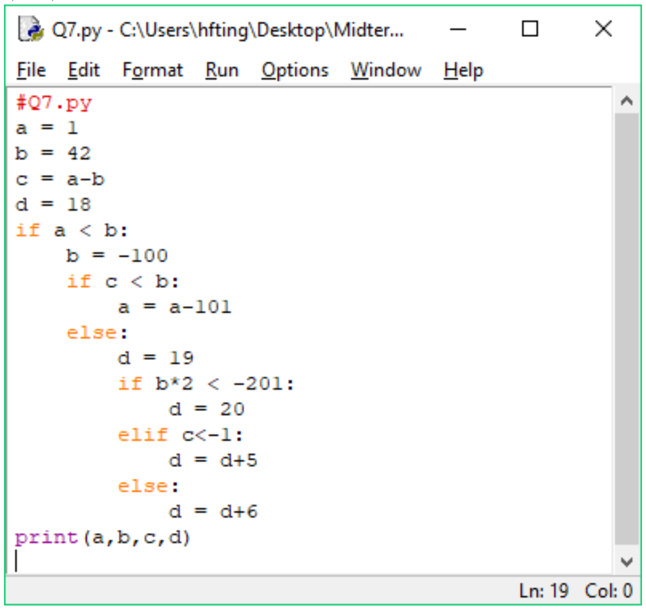
6. (5%)


Q6.py - C:\Users\hfting\Desktop\Midterm\Q6.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 3 Col: 0

```
#Q6.py
print(list(map(lambda x,y: x+y, map(int,'13579'),map(int,'02468'))))
```

[1, 5, 9, 13, 17]

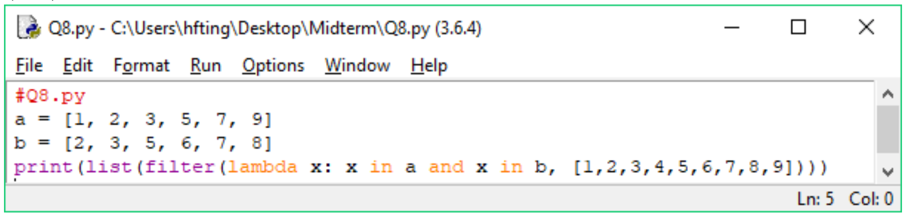
7. (5%)


Q7.py - C:\Users\hfting\Desktop\Midter...
File Edit Format Run Options Window Help
Ln: 19 Col: 0

```
#Q7.py
a = 1
b = 42
c = a-b
d = 18
if a < b:
    b = -100
    if c < b:
        a = a-101
    else:
        d = 19
        if b*2 < -201:
            d = 20
        elif c<-1:
            d = d+5
        else:
            d = d+6
print(a,b,c,d)
```

1 -100 -41 24

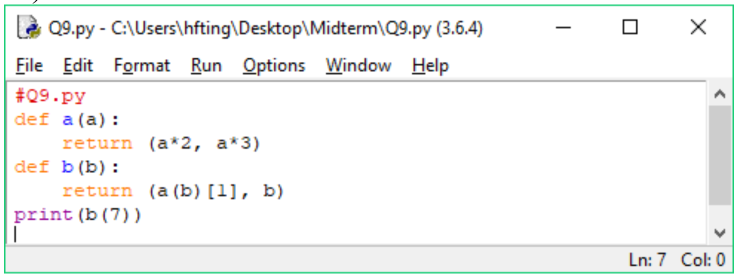
8. (5%)


Q8.py - C:\Users\hfting\Desktop\Midterm\Q8.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 5 Col: 0

```
#Q8.py
a = [1, 2, 3, 5, 7, 9]
b = [2, 3, 5, 6, 7, 8]
print(list(filter(lambda x: x in a and x in b, [1,2,3,4,5,6,7,8,9])))
```

[2, 3, 5, 7]

9. (5%)


Q9.py - C:\Users\hfting\Desktop\Midterm\Q9.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 7 Col: 0

```
#Q9.py
def a(a):
    return (a*2, a*3)
def b(b):
    return (a(b)[1], b)
print(b(7))
```

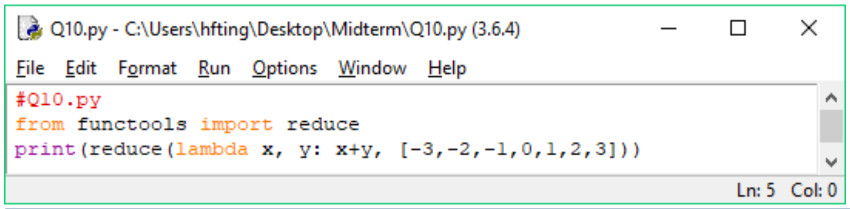
(21, 7)

10. (5%) The following description of Python's function **reduce()** is found on the internet:

The **reduce(fun,seq)** function is used to **apply a particular function passed in its argument to all of the list elements** mentioned in the sequence passed along. It works as follows:

- **fun** is a function that takes two arguments and return a value, and **seq** is a list of elements.
- At first step, the first two elements of **seq** are picked as the first two arguments of **fun** and its return value is obtained.
- Next step is to apply **fun** to the previously attained result and the next element in **seq** that has not been applied to **fun** and the result is again stored.
- This process continues until all the elements in **seq** is processed in this way.
- The final obtained result will be the return value of **reduce(fun,seq).**

What is the output of the following program?


Q10.py - C:\Users\hfting\Desktop\Midterm\Q10.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 5 Col: 0

```
#Q10.py
from functools import reduce
print(reduce(lambda x, y: x+y, [-3,-2,-1,0,1,2,3]))
```

0

A. (25%) A palindrome is a string of characters that reads the same backward as forward. For example, the strings “madam”, “racecar”, and “emittime” are palindromes. Write a non-recursive function (i.e., a function that does not call itself) `is_palindrome(s)`, which returns 1 if the argument `s` is a palindrome, and 0 otherwise.

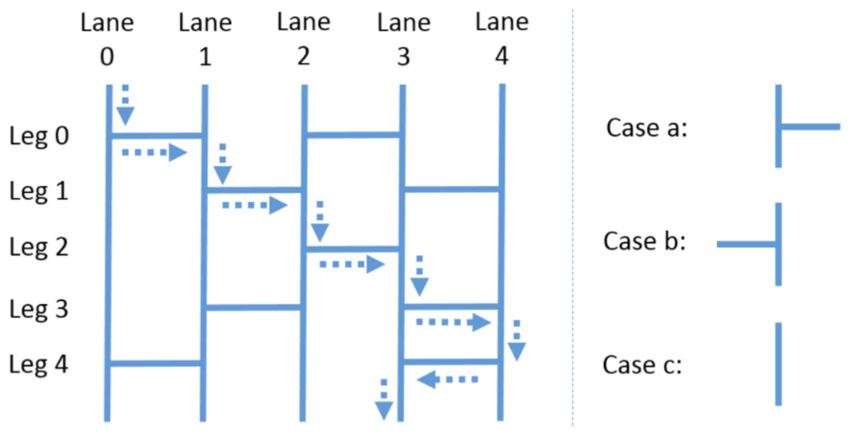
```
def is_def(s):
  return s[0:]==s[-1::-1]
```

Marking Scheme:

- Algorithm / The way of solving the problem is correct but with some errors: 15 Marks
- Algorithm / The way of solving the problem is correct without any errors: 25 Marks

B. **Ghost leg** (25%)

Ghost Leg is a game that distributes things among people. The game has a board, which consists of vertical lines (each is a Lane) with horizontal lines (each is a Leg) connecting two adjacent vertical lines. The following figure gives an example Board of the game.



The rule for playing this game is: choose a Lane (e.g., Lane 0 in the figure), start at top of the Lane and trace downwards. When a Leg is encountered, follow it to get to another Lane and continue downwards. Repeat this procedure until the end of a Lane is reached. For example, choosing Lane 0 ends up at Lane 3 in the example.

We would like to write a Python program to implement the Ghost Leg game.

A board is a `list` of `string`s, with the `i`-th `string` represents the cases of the lanes in leg level `i`. The `j`-th character in the `string` represents the case of the `j`-th lane. The characters can be `a`, `b`, or `c`:

- `a`: Go to the right lane.
- `b`: Go to the left lane.
- `c`: Keep in the same lane.

For example, the board in the figure is represented by:

```
b=['ababc',
   'cabab',
   'ccabc',
   'cabab',
   'abcab']
```

Write a Python function `play(b,n)` that takes `b` as the first input argument which represents the board, `n` as the second argument which represents the starting lane, and returns the resulting lane number of the game. Following are some sample calls of play():

```
b=['ababc','cabab','ccabc','cabab','abcab']
print (play(b, 0))   # Print 3
print (play(b, 3))   # Print 2
```

```
def play(b,n)
   for i in range(len(b)):
     if b[i][n]=='a':
         n = n+1
    elif b[i][n]=='b':
        n = n-1
    return n
```

Marking Scheme:

- Algorithm / The way of solving the problem is correct but with some errors: 15 Marks
- Algorithm / The way of solving the problem is correct without any errors: 25 Marks