Student ID : _____

CITY UNIVERSITY OF HONG KONG

Course code & title : CS1302 Introduction to Computer Programming

Session : Semester B 2021-22

Time allowed : One (1) Hours

This paper has **9** pages (including this page).

Note:

1. Answer ALL the questions in the space provided within each question.

Question	Part 1 (5 marks)	Part 2 (5 marks)	Part 3 (5 marks)	Part 4 (15 marks)	Total (30)
Marks					

2. Do NOT remove the staple or separate the paper.
3. This question paper should NOT be taken away.

Remarks:

For all written code required by the questions:

1. You should give precise Python code with proper programming styles, in particular, appropriate code design, naming and code formatting. Marks may be deducted for redundant or unnecessary code.
2. Unless specifically mentioned, modules/libraries/functions other than built-in functions, math modules are not allowed.
3. Please write down your answer on your A4 or single line paper.

---

*This is a **closed-book** examination.*

No materials or aids are allowed during the whole examination. If any unauthorized materials or aids are found on a candidate during the examination, the candidate will be subject to disciplinary action.

Part 1 (5 marks)

Multiple choice questions (1 mark for each question).
Note that there may be more than 1 correct option for each question, and you must select all the correct options to get the full mark. There is no partial mark such as 0.5. The text in brown color is Python code.

(1) Select all the correct identifier declarations:

A. my-str = "CS1302''
B. _15 = "666"
C. $66 = "hello, world!"
D. type = print

Correct answer: B, D

(2) Select all the correct statements:

A. 2 ** 2 ** 3 is not the equal to (2 ** 2) ** 3 because the exponentiation operator ** is right-associative.
B. x and (1 != x if x else 1) is True regardless of the value of x due to the short circuit evaluation rule of logical and.
C. 0 or 1 and 2 == True is equal to False because True is not even an integer.
D. True or False and True is not equal to (True or False) and True because or is left-associative.

Correct answer: A

(3) Which statements below are correct in Python?

A. Boolean operators have lower precedence than comparison operators.
B. Comparison operators are non-associative.
C. Conditional construct is a form of code reuse.
D. All the comparison operators have the same precedence lower than that of + and -

Correct answer: A, B, D

(4) Which one of the following functions would not return any explicit value? Choose the best answer.

A. A function that checks whether the input number contains more than 5 digits
B. A function that prints integers from 1 to 100
C. A function that returns a random integer from 1 to 100
D. A function that converts an uppercase letter to lowercase

Correct answer: B

(5) What will a function return if the function does not have an explicit return statement?
A. None
B. 0
C. False
D. NaN

Correct answer: A

Part 2 (5 marks)

Please give the output of the following three programs in the correct format.

Program 1 (1 mark):

```
i = 0

while i < 3:
    print(i)
    i= i+1
    print(i+1)
```

Output:

```
0
2
1
3
2
4
```

Program 2 (1 mark):

```
i = 10

for i in range(6):
    if i == 3:
        pass
    elif i == 4:
        continue
    elif i == 5:
        break
    print(i)
else:
    print("End of loop")
```

Output:

```
0
1
2
3
```

Program 3 (1 mark):

```
a = 4
b = True
c = 6
d = False

print(a>b<=c>d)
```

Output:

```
True
```

Program 4 (2 marks):

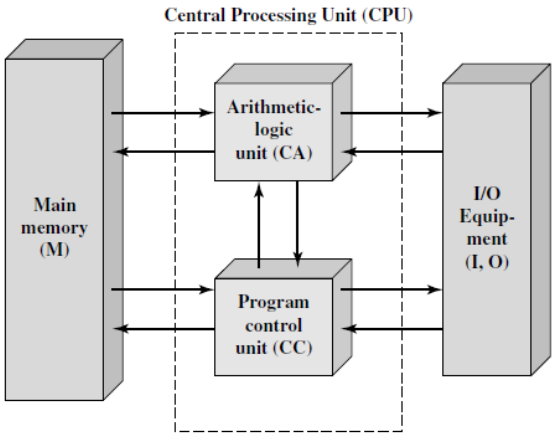
```
print(True and False or True and not False or True)
print(False or False and True and not False or True and False)
```

Output:

```
True
False
```

Part 3 (5 marks)

1. Please draw the Von Neumann architecture and explain the functions of each component (2 marks).



Almost all computers are based on Von Neumann architecture, which consists of the following three main components:

- Input/output device: read information into or out of the memory unit upon command from the CPU.
  - Central Processing Unit (CPU)
  - Arithmetic and Logic Unit (ALU): performs arithmetics like a calculator (but for binary numbers)
- Control Unit (CU): directs the operations of the processor in executing a program.
- Main memory: stores both data and instructions

2. Please complete the following table which converts the same number in different representation systems (0.5 mark for each blank).

Binary (base-2)	Decimal (base-10)	Undecimal (base-11)
101011		
	65	
		2X

Binary (base-2)	Decimal (base-10)	Undecimal (base-11)
101011	43	3X
1000001	65	5X
100000	32	2X

Part 4 (15 marks)

Please complete the following programming questions (3 marks for each question)

1. Define a function odd_numbers_below that
- takes a non-negative integer argument n and
- prints all odd numbers strictly smaller than n in ascending order in one line but separated by a space.

For example:

Test	Result
odd_numbers_below(6)	1 3 5
odd_numbers_below(9)	1 3 5 7
odd_numbers_below(10)	1 3 5 7 9

Answer:

```
def odd_numbers_below(n):
   for i in range(n):
         if i % 2: print(i, end=' ')
```

2. Define a function display_triangle that

- takes a strictly positive number n and
- prints n lines where the (i)-th line consists of (i) stars.

For example:

Test	Result
display_triangle(1)	*
display_triangle(4)	* ** *** ****

Answer:

```
def display_triangle(n):
   for i in range(1,n+1):
         print('*' * i)
```

3. Define a function gcd that

- takes two strictly positive integers x and y as arguments, and
- returns their greatest common divisor, i.e., the largest positive integer that divides both x and y.

For example:

Test	Result
print('{}'.format(gcd(2,3)))	1
print('{}'.format(gcd(2,4)))	2
print('{}'.format(gcd(6,9)))	3

Answer:

```
def gcd(x,y):
   for divisor in range(min(x,y)+1,0,-1):
         if x%divisor == y%divisor == 0:
               return divisor
```

4. If the infection of Omicron is uncontrolled, it will continue spreading in the community.

The expected growth of the number of infection cases can be calculated based on a *spreading factor f* :

> Given that there are *k* cases on a day, the number of new cases on the next day is *k*f*, rounded to the nearest integer

Your task:
Complete the function infect given below that lists the estimated numbers of accumulated infection cases for a number of days based on a range of infection factors with a given interval for the factors.

The function should take in and apply the following arguments:

- days: the count of days,
- f1, f2: the starting and ending values of the range of infection factors,
- df: the interval between the infection factors

Note: - f1, f2, and df have at most 1 decimal digit.
- NO need to take care of the widths of the columns and alignment of the contents in the output.

Given framework and test case:

```
import math

def infect(days, f1, f2, df):
     # your code below









# Testing
infect(8, 3.0, 5.0, 0.2)
```

```
import math

def infect(days, f1, f2, df):
    print("Factor", end="")
    for i in range(0, days + 1):
        print("  Day " + str(i), end=" ")

    print()

    f = float(f1)
    while f <f2+0.01:
        print(round(f,1), end="  ")
        count = 1
        for i in range(0, days + 1):
            print(count, end=" ")
            count += round(count * f)
        print()
        f += df


# infect(3, 1, 8, 1)
infect(10, 3, 5, 0.2)
```

Sample Output:

```
Factor    Day 0    Day 1    Day 2    Day 3    Day 4    Day 5    Day 6    Day 7    Day 8
 3.0        1        4       16       64      256      1024      4096     16384    65536
 3.2        1        4       17       71      298      1252      5258     22084    92753
 3.4        1        4       18       79      348      1531      6736     29638    130407
 3.6        1        5       23      106      488      2245     10327     47504    218518
 3.8        1        5       24      115      552      2650     12720     61056    293069
 4.0        1        5       25      125      625      3125     15625     78125    390625
 4.2        1        5       26      135      702      3650     18980     98696    513219
 4.4        1        5       27      146      788      4255     22977     124076   670010
 4.6        1        6       34      190     1064      5958     33365     186844   1046326
 4.8        1        6       35      203     1177      6827     39597     229663   1332045
 5.0        1        6       36      216     1296      7776     46656     279936   1679616
```

5. You are given a module Days.py that provides two functions as explained in the comments:

```
import random

def int_to_day(i):
    '''
    Accept an input integer i within the range 0 to 6,
       where 0 means Sunday, 1 means Monday, and so on.
    Return a string which is the name of the corresponding day for i
       E.g. int_to_day(0) returns "Sun", int_to_day(1) returns "Mon" and so on.
    '''
    return ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"][i]


def day_to_int(s):
    '''
    Accept an input string s which is the name of a day in the week: E.g. "Thu"
    Return an integer for the day such that 0 means Sunday, 1 means Monday, etc.
       E.g. day_to_int("Sun") returns 0, day_to_int("Mon") returns 1 and so on.
    '''
    return ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"].index(s)
```

Your task:
Write a program that asks the user to type two days (E.g., "Mon", "Sun") and prints out a sequence of days starting from the first day and ending when the second day is reached.

For example:

User's inputs	Output
Mon Tue	Mon => Tue
Wed Sat	Wed => Thu => Fri => Sat
Fri Tue	Fri => Sat => Sun => Mon => Tue
Fri Fri	Fri => Sat => Sun => Mon => Tue => Wed => Thu => Fri

Note: You have to import the functions from Days.py and make good use of the functions in your solution.

Answer:

```
from Days import day_to_int, int_to_day

d1 = day_to_int(input("First day: "))
d2 = day_to_int(input("Last day: "))

d = d1
print(int_to_day(d), end=" ")
d = (d + 1) % 7

while d != d2:
    print("=>", int_to_day(d), end=" ")
    d = (d + 1) % 7

print("=>", int_to_day(d2))
```

-- END OF PAPER --