COMP1001 – Problem solving in IT
Final Examination

Please read the following instructions carefully.

1. Please put away your phones and calculators.
2. If you need to go to bathroom, please raise your hand and get the approval before going.
3. You may use books, notes and any hardcopy materials, but you cannot share them with others.
4. You will be given three hours to solve seven problems. The full mark of this paper is 160:
    - Q1: 30 marks
    - Q2: 20 marks
    - Q3: 20 marks
    - Q4: 20 marks
    - Q5: 20 marks
    - Q6: 20 marks
    - Q7: 30 marks
5. You can only use IDLE and notepad++ for editing and running your program.
6. Submission instructions:
    a. Name all files as "<Student ID>_<Question Number>.py". For example, "12345768d_Q1.py".
    b. At the beginning of your code, type your name and student ID. For example,
        # Name: Chan Tai Man
        # Student ID: 12345678d
    c. There is a submission link for each question under COMP1001 Exam in https://submit.comp.polyu.edu.hk. Submit your .py file for each question to the corresponding submission box or no mark will be given.
    d. After completing a question, you should submit it right away.
7. Before you leave, please MAKE SURE that you have submitted all your .py files. You will bear the sole responsibility if you fail to do so.
8. If you choose to leave before the official end time, please sign out on an attendance list.

1. [30 marks] (Generating a sequence of triangles) In the pre-exam test, you have generated three different types of triangles using the function `list_sym(max, sym)`. The codes are given below.

```
def list_sym(max, sym):
     slist = []
     for i in range(max+1):
          slist.append(i*sym)
     return slist[1:]

def main():
     sym = input("Please enter a symbol: ")
     max = eval(input("Please enter the maximum number of symbols: "))
     choice = input("Please specify how to print the triangle: \
                         \n\t '<' for right-angled triangle \
                         \n\t '>' for left-angled triangle \
                         \n\t '^' for two-equal-sided triangle\n")
     alist = list_sym(max, sym)
     for i in alist:
          if choice == "<":
               print("{:<}\n".format(i),end="")
          elif choice == ">":
               print("{:>{width}}\n".format(i, width = max), end="")
          elif choice == "^":
               if len(i)%2 > 0:
                    print("{:^{width}}".format(i, width = max), end="")
               else:
                    print()
          else:
               print("Sorry, this choice is not valid.\n")
               break
```

In this question, you are asked to generate a sequence of triangles. Therefore, your program shall ask user to also enter the number of triangles to be printed out. If the number is 0, nothing, including blank lines, will be printed. Note from the sample outputs below that there is no space between adjacent triangles.

```
Please enter a symbol: *
Please enter the maximum number of symbols: 6
Please enter the number of triangles: 6
Please specify how to print the triangle:
         '<' for right-angled triangle
         '>' for left-angled triangle
         '^' for two-equal-sided triangle
<
*      *      *      *      *      *
**     **     **     **     **     **
***    ***    ***    ***    ***    ***
****   ****   ****   ****   ****   ****
***** ***** ***** ***** ***** *****
**********************************
```

```
Please enter a symbol: &
Please enter the maximum number of symbols: 6
Please enter the number of triangles: 6
Please specify how to print the triangle:
         '<' for right-angled triangle
         '>' for left-angled triangle
         '^' for two-equal-sided triangle
>
     &      &      &      &      &      &
    &&     &&     &&     &&     &&     &&
   &&&    &&&    &&&    &&&    &&&    &&&
  &&&&   &&&&   &&&&   &&&&   &&&&   &&&&
 &&&&& &&&&& &&&&& &&&&& &&&&& &&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

```
Please enter a symbol: %
Please enter the maximum number of symbols: 10
Please enter the number of triangles: 8
Please specify how to print the triangle:
         '<' for right-angled triangle
         '>' for left-angled triangle
         '^' for two-equal-sided triangle
^
     %        %        %        %        %        %        %        %
    %%%      %%%      %%%      %%%      %%%      %%%      %%%      %%%
   %%%%%    %%%%%    %%%%%    %%%%%    %%%%%    %%%%%    %%%%%    %%%%%
  %%%%%%%  %%%%%%%  %%%%%%%  %%%%%%%  %%%%%%%  %%%%%%%  %%%%%%%  %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2. [20 marks] (Catching input errors) You are given a simple function below, where both `min` and `max` are integers, and `opr` is an operator which could be `"+"`, `"-"`, `"*"`, and `"/"`.

```
def func(min, max, opr):
    if opr == "+":
        return max + min
    elif opr == "-":
        return max - min
    elif opr == "*":
        return max * min
    elif opr == "/":
        return max / min
```

Enhance this function, so that it will raise four types of errors: (1) If `max` and `min` are not both integers, it will raise a `TypeError`, (2) if `min` is greater than `max`, it will raise a `ValueError`, (3) if the operator is `"/"` and `min` is 0, it will raise a `ZeroDivisionError`, and (4) if opr is not supported, it will raise a ValueError.

Write also a `main()` that will use the `try-except` block to test out the ten cases below. You may use a `try-except` block to test each individual case and output the information below. You only need to produce the right column of information in a correct order.

```
The result for func(10,20,'+'):   30
The result for func(20,10,'+'):   The min must not be greater than the max.
The result for func(20,10):       func() missing 1 required positional argument: 'opr'
The result for func(20, '+'):     func() missing 1 required positional argument: 'opr'
The result for func('a','b','-'): unsupported operand type(s) for -: 'str' and 'str'
The result for func(10,20,'-'):   10
The result for func(10,20,'*'):   200
The result for func(10,20,'/'):   2.0
The result for func(0,20,'/'):    Min cannot be zero for this operation.
The result for func(10,20,'^'):   ('The operator', '^', 'is not supported.')
```

3. [20 marks] A famous formula for estimating π is given below:

$$\pi \approxeq 768\sqrt{2-\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+1}}}}}}}}$$

Implement it in Python and print out both the estimated π and the actual value of π in 30 decimal places. The correct outputs are:

```
The estimated pi is: 3.14159046323676172107752790907 4.
The actual pi is:    3.14159265358979311599796346854 4.
```

4. [20 marks] (Counting the number of x-words) In the test, you have developed a function to remove punctuations (except for `"-"`), special symbols and numbers from a string:

```
def remove_punc(inStr):
    listStr = []
    for char in inStr:
        if char.isalpha() == False and char != "-" and char != " ":
            listStr += ""
        else:
            listStr += char
    return "".join(listStr)
```

Write a function called `word_length(fileName, letter)` that accepts a filename `fileName` and a letter `letter`, and returns the counting of the lengths of the words started with `letter` in `filename`. The word counting is case-insensitive (i.e., "apple" and "Apple" will be counted under letter "a"). Moreover, the word counting will begin only after the punctuations (except for `"-"`), special symbols and numbers are removed from a string using `remove_punc(inStr)`.

Write also a `main()` to test `word_length(fileName, letter)` on a file called `plaintext.txt` available from the Moodle submission site. The program will print the statistics of the word length for "a", "b", …, "z", like the one below. You can skip printing for a letter if the file contains no words started with the letter.

```
For letter a:
Length  Freq. in number   Freq. in %
1       97                29.6636
2       43                13.1498
3       105               32.1101
4       11                3.3639
5       20                6.1162
6       22                6.7278
7       15                4.5872
8       5                 1.5291
9       4                 1.2232
10      3                 0.9174
11      0                 0.0000
12      2                 0.6116
```

```
For letter b:
Length  Freq. in number   Freq. in %
1       14                10.3704
2       39                28.8889
3       19                14.0741
4       6                 4.4444
5       10                7.4074
6       12                8.8889
7       20                14.8148
8       8                 5.9259
9       4                 2.9630
10      3                 2.2222
```

```
...
```

```
For letter y:
Length  Freq. in number   Freq. in %
1       0                 0.0000
2       0                 0.0000
3       53                81.5385
4       7                 10.7692
5       3                 4.6154
6       0                 0.0000
7       0                 0.0000
8       2                 3.0769
```

5. [20 marks] (Adding two digits from the addition table) In the test, you have developed a function for the addition lookup table below.

```
def addTable():
    table = []
    for i in range(10):
        row = []
        for j in range(10):
            if len(str(i+j)) == 1:
                row.append("0"+str(i+j))
            else:
                row.append(str(i+j))
        table.append(row)

    return table
```

	0	1	2	3	4	5	6	7	8	9
0	00	01	02	03	04	05	06	07	08	09
1	01	02	03	04	05	06	07	08	09	10
2	02	03	04	05	06	07	08	09	10	11
3	03	04	05	06	07	08	09	10	11	12
4	04	05	06	07	08	09	10	11	12	13
5	05	06	07	08	09	10	11	12	13	14
6	06	07	08	09	10	11	12	13	14	15
7	07	08	09	10	11	12	13	14	15	16
8	08	09	10	11	12	13	14	15	16	17
9	09	10	11	12	13	14	15	16	17	18

Write a function `proc0(table, number1, number2)`, where `table` is the addition table returned by `addTable()`, and `number1` and `number2` are single-digit numbers (i.e., 0, 1, 2, …, 9). The function returns a string of the two-digit result.

Write also a `main()` that asks user for two digits and uses `proc0(table, number1, number2)` to return the result which must be printed in integer. The program repeats asking user for inputs until at least one input is a negative number.

```
Please enter two digits separated by a comma.
Enter a negative digit to terminate: 1,2
        The result of 1+2 is:  3
Please enter two digits separated by a comma.
Enter a negative digit to terminate: 0,0
        The result of 0+0 is:  0
Please enter two digits separated by a comma.
Enter a negative digit to terminate: 9,9
        The result of 9+9 is:  18
Please enter two digits separated by a comma.
Enter a negative digit to terminate: 4,5
        The result of 4+5 is:  9
Please enter two digits separated by a comma.
Enter a negative digit to terminate: -1,0
        Thank you for using the program.
```

6. [20 marks] (Adding three digits from the addition table) Recall from our first class that we can compute the addition of three digits based on PROC0 (i.e., `proc0()` in the last question) for an addition of two digits using the algorithm PROC1.

Implement the PROC1 in a function `proc1(table, number1, number2, number3)` which returns a string of the two-digit result. If you cannot implement `proc0(table, number1, number2)` using the `addTable()` function in the last question, you could use the `addTable()` below, a hard-coded function, to implement function `proc1()`.

Similar to the last question, write also a `main()` to test `proc1(table, number1, number2, number3)` using the cases below.

Procedure PROC1:

You are given three digits as input, a, t, and b.
You will return two digits as output, c and s.

1. Call PROC0 on t and b, and let u and v be the answers returned. (The u is the left digit returned by PROC0, and the v is the right one.)
2. Call PROC0 on a and v, and let u' and v' be the answers returned. (The u' is the left digit returned, and the v' is the right one.)
3. If $u = u' = 0$, then return with $c = 0$ and $s = v'$.
   If $u = u' = 1$, then return with $c = 2$ and $s = v'$.
   Otherwise, return with $c = 1$ and $s = v'$.


```
Please enter three digits separated by a comma.
Enter a negative digit to terminate: 1,2,3
        The result of 1+2+3 is:  6
Please enter three digits separated by a comma.
Enter a negative digit to terminate: 0,0,0
        The result of 0+0+0 is:  0
Please enter three digits separated by a comma.
Enter a negative digit to terminate: 9,9,9
        The result of 9+9+9 is:  27
Please enter three digits separated by a comma.
Enter a negative digit to terminate: 4,5,6
        The result of 4+5+6 is:  15
Please enter three digits separated by a comma.
Enter a negative digit to terminate: -1,0,0
        Thank you for using the program.
```

```
def addTable():
    table = [['00', '01', '02', '03', '04', '05', '06', '07', '08', '09'],
             ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10'],
             ['02', '03', '04', '05', '06', '07', '08', '09', '10', '11'],
             ['03', '04', '05', '06', '07', '08', '09', '10', '11', '12'],
             ['04', '05', '06', '07', '08', '09', '10', '11', '12', '13'],
             ['05', '06', '07', '08', '09', '10', '11', '12', '13', '14'],
             ['06', '07', '08', '09', '10', '11', '12', '13', '14', '15'],
             ['07', '08', '09', '10', '11', '12', '13', '14', '15', '16'],
             ['08', '09', '10', '11', '12', '13', '14', '15', '16', '17'],
             ['09', '10', '11', '12', '13', '14', '15', '16', '17', '18']]
    return table
```

7. [30 marks] (Revisiting the multiplication for Rational) Recall from our lecture that we have developed a class for rational numbers below:

```
def gcd(a, b):
     while a % b != 0:
          a, b = b, a % b
     return b

def lcm(a, b):
     return (a*b)//gcd(a,b)

class Rational(object):
     def __init__(self, numer, denom=1):
          ''' Rational with numerator and denominator, denominator defaults to 1'''
          if denom == 0:
               raise ZeroDivisionError("The denominator cannot be zero.")
          self.numer = numer
          self.denom = denom

     def __str__(self):
          ''' String representation for printing '''
          return str(self.numer)+"/"+str(self.denom)

     def __repr__(self):
          ''' Used in interpreter. Call __str__for now '''
          return self.__str__()

     def __add__(self, param_Rational):
          ''' Add two Rationals '''
          the_lcm = lcm(self.denom, param_Rational.denom)
          numerator_sum = (the_lcm * self.numer / self.denom) + (the_lcm * param_Rational.numer / param_Rational.denom)
          return Rational(int(numerator_sum), the_lcm)

     def reduce_Rational(self):
          ''' Return the reduced fractional value as a Rational. '''
          the_gcd = gcd(self.numer, self.denom)
             return Rational(self.numer//the_gcd, self.denom//the_gcd)
```

In this question, you are asked to perform four modifications:

a. Instead of expressing a rational number as 3/2, a better way is 1½ . Note that 1½ is the same as 1 + ½. You therefore will add an additional instance variable to the class called `whole`. Therefore `self.whole = 1`, `self.numer = 1` and `self.denom = 2` for this example. In the `__init__()` function, you will make `self.whole = 0` by default.
b. Modify the `reduce_Rational()` function, so that the rational number will be reduced to a whole number and a fractional part in which the numerator is always less than the denominator. For example 12/8 is simplified to 1½ .
c. Modify the `__add__()` function, so that it will add two given `Rational` instances and return the result after being simplified by the new `reduce_Rational()` function.
d. Modify the `__str__()` function, so that `print(x)` and `str(x)`, where `x` is a `Rational` instance, will print it in the format of `x.whole/x.numer/x.denom`. However, if `x.whole` is 0, it will print only `x.numer/x.denom`. Moreover, if `x.denom` is 1, only an integer is printed.

Include a `main()` that prints out the values of the following ten expressions using `print()`.

a. `Rational(1)`
b. `Rational(2,3)`
c. `Rational(1, whole=2)`
d. `Rational(whole=1, numer=1, denom=2)`
e. `Rational(2,3,1)`
f. `x + y`, where `x = Rational(1,2,3)` and `y = Rational(3,2,1)`
g. `x + y`, where `x = Rational(1,2)` and `y = Rational(3,2,1)`
h. `x + y`, where `x = Rational(1)` and `y = Rational(3,2,1)`
i. `x + y`, where `x = Rational(numer=0, whole=1)` and `y = Rational(3,2,1)`
j. `x + y`, where `x = Rational(numer=5, denom=3, whole=1)` and `y = Rational(3,2,1)`

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

END