COMP1001 – Problem Solving in Information Technology
Final Examination

Please read the following instructions carefully.

1. Please put away your phones and calculators. You are allowed to use only the PC for the exam.
2. If you need to go to bathroom, please raise your hand and get the approval before going.
3. You will be given three hours to solve seven problems. The full mark of this paper is 150.
    - Q1: 20 marks
    - Q2: 20 marks
    - Q3: 20 marks
    - Q4: 20 marks (10 marks each for parts (a) and (b))
    - Q5: 20 marks (10 marks each for parts (a) and (b))
    - Q6: 30 marks (15 marks each for parts (a) and (b))
    - Q7: 20 marks
4. Unless stated otherwise, you do not need to handle exceptions in your programs.
5. Under the Announcements of https://submit.comp.polyu.edu.hk, you will find graphics.py, a stack ADT implementation, and other necessary files.
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. Submit your solutions to https://submit.comp.polyu.edu.hk.
    d. After completing a question, you should submit it right away.
7. You cannot leave before the end of the exam.

1. (**Braces balancing**) This problem extends Q2 of A3 by considering three different kinds of braces in an input. Write a Python program that will ask user for a combination of open braces "(", "[", "{" and close braces ")","]","}", and **check whether the braces are balanced**. Some sample outputs are given below.

```
>>>
Please enter the combination of braces: {{})[]
The result is False.
>>> main()
Please enter the combination of braces: ({}[]({}))
The result is True.
>>> main()
Please enter the combination of braces: {{(){[()]}
The result is False.
>>> main()
Please enter the combination of braces: (([]{[]}))[{}()[]]
The result is True.
>>> main()
Please enter the combination of braces: (){}[]
The result is True.
>>> main()
Please enter the combination of braces: ({[]})
The result is True.
>>>
```

2. (**Random integer generation**) **This questions concerns generating a set of random integers.** You are asked to implement `randomInt(min, max, num)` which returns a sorted tuple of `num` integers generated randomly from [`min`, `max`]. **You must catch any possible exception in your implementation.** You must also include a `main()` that can catch the exceptions raised from `randomInt(min, max, num)` and print out the error messages. Moreover, the `main()` must include the test cases below. Note that the test cases intentionally do not include any exception case.

```
>>>
The output for randomInt(10,10,0) is:  ()

The output for randomInt(0,100,101) is:  (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
92, 93, 94, 95, 96, 97, 98, 99, 100)

The output for randomInt(0,100,50) is:  (1, 2, 3, 6, 8, 9, 14, 15, 17, 19, 20, 2
2, 23, 24, 27, 29, 30, 32, 38, 39, 42, 43, 47, 51, 56, 60, 61, 64, 67, 68, 71, 7
2, 73, 74, 76, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 91, 92, 93, 98, 100)
```

Some useful functions in module `random`:

- `random()` chooses a random floating point number in the range [0, 1).
- `randrange(a,b)` chooses an integer in the range [a, b).
- `uniform(a,b)` chooses a random floating-point number in the range [a, b).

3. (**Pascal's Triangle**) A Pascal's Triangle, shown in Figure 1 for 7 levels, is a famous mathematical structure. A Pascal's Triangle of 0 level simply consists of 1. For one level, it consists of 1 and 1. For two levels and above, an algorithm for computing the triangle is shown in Figure 2. Given row *i,* an inner element of row *i* + 1 is given by adding two adjacent numbers on row *i*.

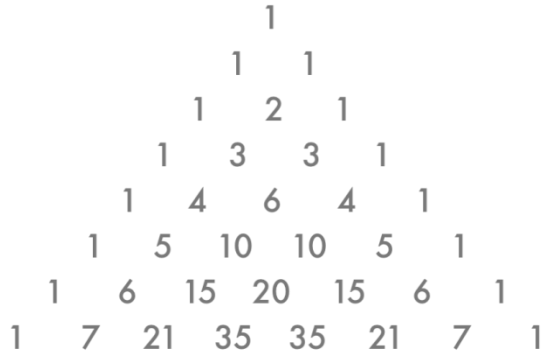

Figure 1: A Pascal's Triangle of 7 levels.

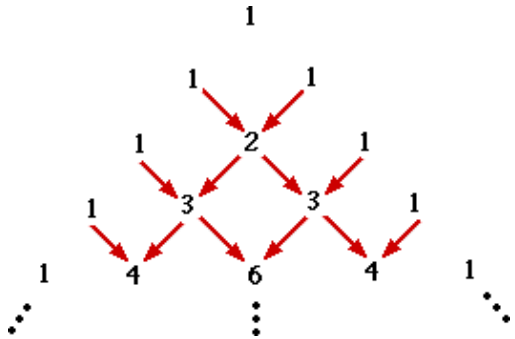

Figure 2: An algorithm for computing a Pascal's Triangle for 4 levels.

Implement a function named `pascalTri(n)` that will print out the rows of numbers for a Pascal's Triangle of *n* levels like the one below (but not the ones in Figures 1 and 2). Include a `main()` that will print out a 10-level Pascal's Triangle as depicted below.

```
>>>
The Pascal's Triangle for 10 levels is:
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
[1, 6, 15, 20, 15, 6, 1]
[1, 7, 21, 35, 35, 21, 7, 1]
[1, 8, 28, 56, 70, 56, 28, 8, 1]
[1, 9, 36, 84, 126, 126, 84, 36, 9, 1]
[1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1]
```

4. **(Another line-art crafting)**
   a) **Write a Python program to generate a square graphical window with the pattern in Figure 3.**

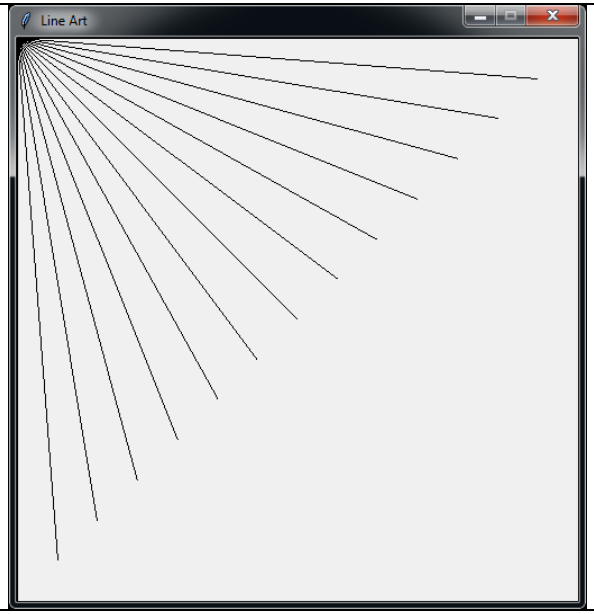

Figure 3: First pattern.

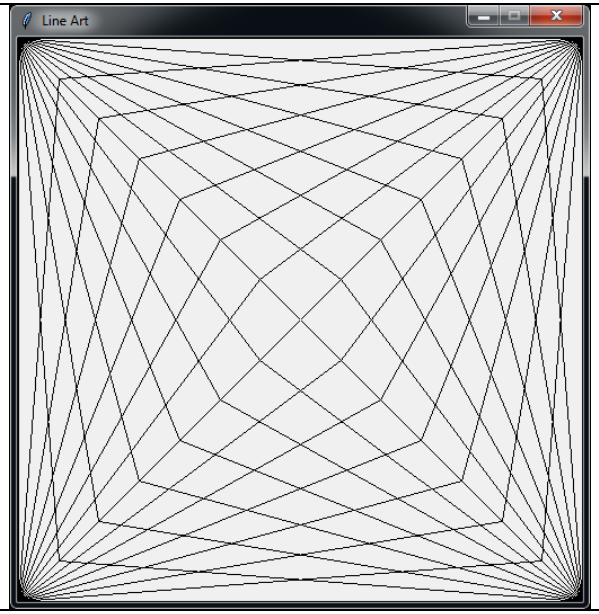

Figure 4: Second pattern.

This design draws lines from the top-left corner, fanning out the lines until they cover the upper-left half of the panel. There are 15 straight lines including the two lines overlapping with the two edges of the window. The dimension of the window is 500 x 500.

b) **Write a program to have lines fan out from all four corners.** Lines from opposite corners should intersect along the middle. The expected pattern is shown in Figure 4.

5. (**Adding the exponentiation operator**) This question concerns **the problem of converting an infix expression to a postfix expression**. In the lecture, we were concerned about *, /, +, and -. In this question, **we also consider the ** operator**.

   a) Before we implement the function `inToPost(inExp)`, we add ** to the table below. **Fill in the missing items in the table below** which is available under the Announcements of https://submit.comp.polyu.edu.hk.

Row: infix char Col: char on the top of the stack	(	+, −	*, /	**	Empty
Identifier	Append to postfix	Append to postfix	Append to postfix	?	Append to postfix
)	Pop: pitch '('	Pop to postfix	Pop to postfix	?	Error
(	Push	Push	Push	?	Push
+, −	Push	Pop to postfix	Pop to postfix	?	Push
*, /	Push	Push	Pop to postfix	?	Push
**	?	?	?	?	?
empty	Error	Pop to postfix	Pop to postfix	?	Done

   b) **Implement the function `inToPost(inExp)`**. Sample outputs are given below.

```
>>>
Please enter an arithmetic expression in which every item is separated by at least a space: a - ( b + c * d ) / e
a b c d * + e / -
>>> main()
Please enter an arithmetic expression in which every item is separated by at least a space: a ** b ** c
a b c ** **
>>> main()
Please enter an arithmetic expression in which every item is separated by at least a space: a * b ** c / d
a b c ** * d /
>>> main()
Please enter an arithmetic expression in which every item is separated by at least a space: ( a * b ) ** ( c + d )
a b * c d + **
```

6. (**Processing student records**) You are given a file containing the student records in a class. By opening `student_record.csv` under the Announcements of https://submit.comp.polyu.edu.hk, you will see that it contains the student names, their IDs, exam marks, lab marks, and assignment marks.

   a) In the first part, **you will implement a function called `recordDict(fileName)` that will take in a file name of the student records, store them in a dictionary, and return the dictionary.** Also include a `main()` that invokes `recordDict(student_record.csv)` and prints the keys and items of the returned dictionary. Below are keys and items for the first few records. Notice that the key consists of the student names and IDs, and the value consists of three dictionaries for the exam marks, lab marks, and assignment marks with keys "Exam", "Lab", and "A", respectively. The values of the lab and assignment dictionaries are lists which contain the marks for a consecutive number of lab/assignment marks. Assume that the labs and assignments are numbered from 1, 2, and so on.

   While names, IDs and exam marks take one data field each, the number of data fields for the lab/assignment marks cannot be determined beforehand. Therefore, **you cannot hardcode the number of labs and assignments in your program**. You will not receive any mark if you do so.

```
>>>
('CHAN Chan', '99188780D')      [{'Exam': '42.5'}, {'Lab': ['5', '5', '5', '9', '10']}, {'A': ['97', '100', '100', '100']}]
('CHAN Ming', '99906277D')      [{'Exam': '28.5'}, {'Lab': ['3', '5', '5', '9', '10']}, {'A': ['97', '80', '95', '100']}]
('CHAN Kwok', '99101419D')      [{'Exam': '32.5'}, {'Lab': ['2', '5', '5', '10', '10']}, {'A': ['90', '95', '100', '100']}]
```

   b) In the second part, **you are asked to implement a function called `readDatabase(aDict,name,ID,item)`**. This function prints the mark for an assessment (i.e., exam, lab, and assignment) specified in *item* for a student whose name and ID are given in *name* and *ID*, respectively. Specifically, the *item* could be "exam", "labN", and "assignN", where *N* is an integer starting from 1, for the N^{th} lab/assignment. The function also needs to pass the dictionary for the student records through the first argument. Moreover, it has to raise errors for a few possible exceptions:
      - KeyError: when the key (*name* and *ID*) is not in the database.
      - IndexError: when *N* is out of range for labs and assignments.
      - ValueError: when *item* is not in the database.

Also include a `main()` that calls the function with *aDict* generated from `recordDict(student_record.csv)` in part (a). Some sample outputs below:

```
>>>
Please enter the name and ID of the student separated by a comma: LAM Ho,99005988D
Please enter the record item (exam, labN or assignN, where N is an integer starting from 1: exam
The marks for exam: 48
>>> main()
Please enter the name and ID of the student separated by a comma: LAM Ho,99005988D
Please enter the record item (exam, labN or assignN, where N is an integer starting from 1: lab4
The marks for lab4: 9
>>> main()
Please enter the name and ID of the student separated by a comma: LAM Ho,99005988D
Please enter the record item (exam, labN or assignN, where N is an integer starting from 1: assign3
The marks for assign3: 100
>>> main()
Please enter the name and ID of the student separated by a comma: LAM Ho,99005988D
Please enter the record item (exam, labN or assignN, where N is an integer starting from 1: exam1
ValueError: The specified assessment is not in the database.
>>> main()
Please enter the name and ID of the student separated by a comma: x,y
Please enter the record item (exam, labN or assignN, where N is an integer starting from 1: exam
KeyError: The specified record is not in the database.
>>> main()
Please enter the name and ID of the student separated by a comma: LAM Ho,99005988D
Please enter the record item (exam, labN or assignN, where N is an integer starting from 1: lab10
IndexError: The specified N is out of range.
>>> main()
Please enter the name and ID of the student separated by a comma: LAM Ho,99005988D
Please enter the record item (exam, labN or assignN, where N is an integer starting from 1: assign20
IndexError: The specified N is out of range.
```

7. (**Partial anonymity**) At the end of my classes, I usually post the continuous marks for the students to check their accuracy. Since they will be posted in public, I have to observe the user privacy. A simple way is to truncate the student IDs to the extent that the truncated IDs are still distinguishable. **You are asked in this question to produce the most truncated IDs for this purpose**. We assume that we will truncate the IDs only from the left hand side.

   Write a program that accepts a csv file of student records in which ID is one of the fields and prints out the truncated IDs. You will use the `student_record.csv` given for the last question. The first few truncated IDs are given below:

780D
419D
277D
944D
895D
409D
942D
605D
675D
252D
158D
927D
679D
513D
639D
691D
…

~~ END ~~