COMP1001 Additional programming exercises

1. [Exam 2017/18] Consider the pseudocode for a function called *eval()* which evaluates an arithmetic expression.

   eval(aList) → *a number*
   input: aList contains either operators (+, –, ×, and /) or operands (numbers or variables).
   output: a number

   create a stack called aStack
   for item in aList do
   if item is an operand, then
   aStack.push(item)
   else
   opr2 = aStack.pop()
   opr1 = aStack.pop()
   result ← cal(item, opr1, opr2)
   aStack.push(result)

   return aStack.pop()

   For *cal(item, opr1, opr2)*, *item* could be "+", "–", "×", and "/". *cal()* evaluates "*opr1 item opr2*" and returns the value. For example, *cal(*"/", 4, 2) evaluates 4/2 and returns 2.0.

   a. [2 marks] If *aList* = [1, 2, 3, "×", "+"], what would be the value returned by *eval(aList)*? Explain your answer.
   b. [3 marks] If *aList* = [1, 2, "+", 3, "×", 4, "+"], what would be the value returned by *eval(aList)*? Explain your answer.
   c. [5 marks] If *aList* = [6, 3, "/", 2, 4, "+", "×", 5, "/"], what would be the value returned by *eval(aList)*? Explain your answer.

2. (Exam, 2017/18) The following program implements a sorting algorithm called merge sort. The only missing part is the implementation of the `merge()` function. Given two lists of data sorted in ascending order. The function returns a list which contains all the data items from the two lists and they are sorted in ascending order. For example, merge([1, 4, 6, 10], [5, 7, 8, 11, 12]) returns [1, 4, 5, 6, 7, 8, 10, 11, 12]. Without using any functions from built-in or imported modules, other than the function `len()`, implement the function `merge()`.

```
def merge(lefthalf, righthalf):
    #
    # Implement your code here
    #

def mergesort(x):
    if len(x) == 0 or len(x) == 1:
        return x
    else:
        middle = int(len(x)/2)
        a = mergesort(x[:middle])
        b = mergesort(x[middle:])
```

```
        return merge(a,b)

def main():
    aList = [10, 5, 2, 9, 6, 3, 4, 8, 1, 7]
    print(mergesort(aList))
    list1 = [1, 4, 6, 10]
    list2 = [5, 7, 8, 11, 12]
    print(merge(list1, list2))

main()
```

Expected output of the program:

```
>>> ================================ RESTART ================================
>>>
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 4, 5, 6, 7, 8, 10, 11, 12]
>>>
```

3. [Test 2017/18] Consider the function below that computes the sum of 1, 2, …, and i for $i = 1, \ldots, n$.

```
def sum(n):
    print("{0:<8}{1:<8}".format("n","sum"))
    for i in range(1,n+1):
        result = 0
        for j in range(1,i+1):
            result = result + j
        print("{0:<8d}{1:<8d}".format(i,result))
```

Below are the outputs for $n = 10$.

```
1       1
2       3
3       6
4       10
5       15
6       21
7       28
8       36
9       45
10      55
```

   a. [5 marks] Notice that this program uses a brute-force approach to find the sums. For this problem we use the number of additions performed to measure the time complexity. What is the number of additions performed by this function for $n > 0$?

   b. [5 marks] How can you improve the time complexity of this function to $O(n)$? Write down the program by hand without the print statements.

4. [Test 2017/18] A palindrome is a word, phrase, number, or other sequence of characters which reads the same forward or backward. Write a Python function called *isPalindrome(aString)* to determine whether *aString* is a palindrome or not. If it is, it will return `True`, else `False`. Here

we consider only a word or a number. **You must use stack for the implementation of this function**. In *main()*, test the six cases below and the expected answers are given on the right hand side of "→." Name the file as `[your studentID]_Q3.py`.

- isPalindrome("Rocky") → `False`
- isPalindrome("Hannah") → `True`
- isPalindrome("madam") → `True`
- isPalindrome("Hanna") → `False`
- isPalindrome("123321") → `True`
- isPalindrome("123456") → `False`

5. [Test 2017/18] The following program generates the calendar of 2018. On input a month, the program displays the calendar of that month. For your information, 1st of January in 2018 is Monday and 2018 is not a leap year. Implement the function `determine1stDayofMonth(m, days)`. Assume user input is always correct and no data validation is required.

```
days = [ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 ]
# 0 represents Monday, 1 represents Tuesday and so on.
FIRST_DAY_JAN = 0;

def main():

    m = int(input("Please enter a month (1 - 12) or -1 to end:
"))

    while m != -1:

        # 0 represents Monday, 1 represents Tuesday and so on.
        startingDay = determine1stDayofMonth(m, days)

        # print the calendar
        print("M   T   W   T   F   S   S")
        for i in range(startingDay):
            x = " "
            print("{0:<4}".format(x), end="")
        j = startingDay
        for i in range(1, days[m - 1] + 1):
            if j % 7 == 0 and i != 1:
                j = 0
                print()

            print("{0:<4}".format(i), end="")
            j = j + 1
```

```
        print()
        m = int(input("Please enter a month (1 - 12) or -1 to
end: "))

    print("Bye!");

def determine1stDayofMonth(m, days):
    #
    # Implement your code here
    #

main()
```

Sample input and output:

```
>>> ================================ RESTART ================================
>>>
Please enter a month (1 - 12) or -1 to end: 1
M   T   W   T   F   S   S
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
Please enter a month (1 - 12) or -1 to end: 5
M   T   W   T   F   S   S
    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
Please enter a month (1 - 12) or -1 to end: 9
M   T   W   T   F   S   S
                    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
Please enter a month (1 - 12) or -1 to end: -1
Bye!
>>>
```

Name the file as [your studentID]_Q6.py.

6. [Test 2017/18] Recall that a palindrome is a word, phrase, number, or other sequence of characters which reads the same forward or backward. Write a Python function called *createPalindrome(aString)* to generate a palindrome. *aString* could be a number or string, including a null string. The palindrome is formed by interleaving *aString* and the reverse order of the characters in *aString*. For example,

- *aString* = "abc"
- reverse of *aString* = "cba"
- output = "acbbca"

You must use stack for the implementation of this function. In *main()*, test the six cases below and the expected answers are given on the right hand side of →. Name the file as `[your studentID]_Q3.py`.

- createPalindrome("abc") → "acbbca"
- createPalindrome("abcd") → "adbccbda"
- createPalindrome("123456") → "162534435261"
- createPalindrome("654321") → "615243342516"
- createPalindrome("a") → "aa"
- createPalindrome("") → ""

7. [Test 2017/18] The following program accepts a list of votes. Assume there are three candidates (i.e., 1, 2 and 3) only. Assume user input is always correct and no data validation is required.

```
def main():

    inputV = input("Please enter a list of votes: ")
    listV = inputV.split()

    # count stores the number of votes of each candidate
    count = [0, 0, 0]
    for i in listV:
        count[int(i)-1] = count[int(i)-1] + 1

    printHistogram(count)

def printHistogram(count):
    #
    # Implement your code here
    #

main()
```

Implement the function, `printHistogram(count)`, to print a histogram in the following way:

```
Please enter a list of votes: 1 2 3 3 2 1 1 1 2 2 2
Here is the histogram:
    *
*   *
*   *
*   *   *
*   *   *
1   2   3
>>>
```

Name the file as [your studentID]_Q6.py.

8. (Exam 2016/17) 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.
>>>
```

9. [Exam 2016/17] 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.
```

10. [Exam 2016/17] 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
    …