NATIONAL UNIVERSITY OF SINGAPORE

ANSWER

SCHOOL OF COMPUTING
SEMESTER II AY2019/2020

MIDTERM ASSESSMENT
IT5003: DATA STRUCTURES AND ALGORITHMS

November 2019
Time Allowed: 1.5 Hours

STUDENT NUMBER:	A	0						

INSTRUCTIONS TO CANDIDATES:

1. Write your student number in the space provided above using a **PEN**.

2. This examination paper consists **SIX (6) questions** and comprises **TEN(10)** printed pages including this front page.

3. Answer all questions directly in the space given after each question. You can **use PENCIL or PEN.**

4. Marks allocated to each question are indicated. Total marks for the paper is **40**.

5. This is a OPEN BOOK assessment. Non-programmable Calculators are allowed. Other electronic devices, e.g. laptop, smart phone are **prohibited**.

Questions	Possible	Marks
Q1	6	
Q2	8	
Q3	6	
Q4	6	
Q5	8	
Q6	6	
Total	**40**	

Question 1 (6 marks)

Suppose you are given the following **Bag ADT**:

```
class Bag:
    def exists(self, item):
    """ return True if item is in the Bag. False otherwise"""

    def addItem(self, newItem):
    """ add a [newItem] into the Bag. If [newItem] already exists, this method
    does nothing."""

    def addCopy(self, item):
     """ Increase the count of the same item in the Bag. If [item] does not exists,
    this method does nothing. """

    def  print(self):
    """ Print out the each item in the Bag, along with the number of copies for
    each of them. """
```

If we want to count the frequency of word (e.g. how many "the"? how many "a"? etc), show how you can easily use the Bag ADT to accomplish this task.

You can assume the input sL is a list of words in the essay.

```
def wordFrequency( sL ):
      wordBag = Bag()

      for word in sL:
           if not wordBag.exist( sL ):
                  wordBag.addItem( word )
           else:
                  wordBag.addCopy( word )

      wordBag.print()
```

Question 2 (8 marks)

The **median-of-3 quicksort** introduces a small change to the partitioning algorithm. Given an array **A** with range **[i…j]**, instead of choosing the first element **A[i]** as the pivot, this approach pick the **median element** between **A[i]**, **A[j]** and **A[m]**. The **m** index is simply $\left\lfloor \frac{i+j}{2} \right\rfloor$. For example, if A is [**7**, 3, **2**, 8, 9, **5**], then **5** is chosen as the pivot as it is the median between **7** (A[0]), **5**(A[5]) and **2** (A[$\left\lfloor \frac{0+5}{2} \right\rfloor$]).

a. Show how can you add some code to the partition function to implement this idea while reusing **all of the original partitioning code**. **Restriction:** Your additional code can only swap the **median element** with **another element**, i.e. all other elements must stay in the original location. **(3 marks)**

```
def partition(array, i, j ):
"""Implement the median-of-3 partitioning. """

#ans: Key idea: move median element into array[i]
   p = i
   m = (i+j) // 2
   if array[i] > array[m]:
       i, m = m, i
   if array[m] > array[j]:
       j, m = m, j
   if array[i] > array[m]:
       i, m = m, i
   swapElement(array, p, m)




    #original partitioning code below
    pivot = array[i]
    middle = i
     ……. other code in original partition can be used without any change……
```

b. Suppose we have a sequence of numbers from 1 to 6 to be sorted by this new median-of-3 quicksort, give an **example sequence** that will result in the **worst case performance**. **(2 marks)**

The unsorted sequence is:

1	4	2	5	6	3

c. Give the complexity of the worst case. Show your working. **(3 marks)**

Worst case:

A N-element partitioning results in 1 element in the left, and N-2 element in the right.

i.e. partition size will shrink by 2 instead of the ideal N/2

Since N-2 ➔ N- 4 ➔ N - 6 ➔ …. ➔ 1 still takes O(N) steps, the worst case is still O(N^2).

Question 3 (6 marks)

In this question, we will consider the issue of **removing the first** matched item **from a sequence of values.** For example, given the sequence {1, 3, **2**, 5, 2, 2} and the target to remove is "2", the result is {1, 3, 5, 2, 2}, where the **first "2"** is removed.

The only difference between the two parts below is the data structure used to store the sequence of values. In part (a), the values are stored in a Python List (i.e. array), in part (b), the values are stored in a Singly linked list. **Give a recursive solution for both parts.** Although pseudo code is allowed, your answer needs to be clear enough to receive full marks. **Each part is worth 3 marks**.

a.

```
def remFirstL( L, target ):
""" Remove the first value in L that matches target. Return a Python List.
    L is a Python List. The original list is returned if there is no match in L.
"""

    if L == []:
        return L

    if L[0] == target:
        return L[1:]
    else:
        return [L[0]] + remFirst( L[1:], target )
```

b.

```
def remFirstR( R, target ):
""" Remove the first value in R that matches target. Return a Linked List.
    R is a Linked List, where each node is a SinglyNode with {item, next} fields.
    The original list is returned if there is no match in R.
"""

  if R == None:
      return None


  if R.item == target:
       return R.next

  R.next = remFirstR( R.next, target)

  return R
```

Question 4 (6 marks)

```
def s( q, result ):
""" Mysterious function. Both [q] and [result] are Queue ADT
   """

   while not q.isEmpty():
       r = q.getFront()

       result.enqueue( r )
       q.dequeue()
       qSize = q.size()

       for i in range( qSize ):
           item = q.getFront()
           if item % r != 0:
               q.enqueue( q.getFront() )
           q.dequeue()
```

a. If we invoke the above function by **`s(myQ, myRes)`**, where **`myQ`** contains values from 2 to 10 (2 is at the front), and **myRes** is empty, answer the following:

i. What is the **number of values** in **`myRes`** after the finishing the for loop **for the first time** **(1 marks)**

1

ii. What is the **number of values** in **`myRes`** **at the end of the function**? **(2 marks)**

4

b. What is the purpose of the function **`s( q, result)`** if we place consecutive and ascending numbers from **2 to N** i.e. [2, 3, 4, ….N-1, N] in **q**? Describe the functionality as accurate as possible. **(3 marks)**

Find all prime numbers between 2 to N and place them in **result** in increasing order

Question 5 (8 marks)

Given **N** integers, a value **x** is known as the **majority number** if there more than $\left\lfloor \frac{N}{2} \right\rfloor$ occurences of **x** among the integers. For example, in [5, 3, 1, 5, 3, 5, 5], 5 is the majority number as there are more than $\left\lfloor \frac{7}{2} \right\rfloor = 3$ 5s among the numbers. Note that it is possible that there is no majority number in a sequence, e.g. [6, 5, 5, 2] has no majority number as there are only two 5s, which is not more than $\left\lfloor \frac{4}{2} \right\rfloor = 3$ occurrences.

We are interested in finding out the majority number given **N unsorted numbers**.

a. Give a O(N lg N) **algorithm** to find the majority number. You can directly use algorithms learned in course so far (e.g. "1. Binary Search N for a number Y, etc"). Indicate the complexity for each major step in your algorithm. **(4 marks)**

ANS:

- Merge sort the numbers = O(N lg N)
- Count occurences of each number, maintain the max count = O(N)

b. Give a **stack algorithm** in O(N) time to find the majority number. [Hint: If we remove two **different** numbers from the sequence, the majority number is still the same as the original sequence.] **(4 marks)**

ANS:

1. First find a possible candidate by using the hint:

```
def getCandidate( L ):
""" L is the list of N numbers """
     cS = Stack()
     for item in L:
     if not cS.isEmpty() and cS.getTop() != item:
          cS.pop()
     else:
          cS.push( item )

     return cS.getTop()
```

Cost = O(N)

2. Use the candidate to go through the array to ensure it is the majority

Cost = O(N)

Question 6 (6 marks)

Suppose you are given a singly linked list **L** and a number **N**. Write a function to split **L** into two linked list: **Ls** where all nodes are **<= N** and **Lb** where all nodes are **> N**. For example, if **L** is 5 → 2 → 7 → 3 → 4 → 9 and N is **5**, then **Ls** is 4→ 3→ 2→ 5 → and **Lb** is 9 → 7. Note that **Ls** and **Lb** contains the original nodes in reverse order.

You are **not allowed to create new linked list node**, i.e. you need to "rewire" the original nodes for **Ls** and **Lb**.

[**Bonus 3 marks**: Instead, give a **recursive version** that returns the **(Ls, Lb)** in the original order. You will receive bonus marks only if the answer is correct. You can change the return statement only if you are attempting the recursive version.]

```
def splitBy( L, N ):
    """ L is the head reference pointing to a singly linked
    list. Each node contains {item, next}. """


    #use Ls, Lb to build the two lists

   #ans
      Ls = Lb = None
      while L != None:
              cur = L
              L = L.next
              if cur.item <= N:
                      cur.next = Ls
                      Ls = cur
               else:
                      cur.next = Lb
                      Lb = cur




      return (Ls, Lb)
```

```
def splitBy( L, N ):
""" L is the head reference pointing to a singly linked list.
Each node contains {item, next}. """

    #Bonus
    if L == None:
        return (None, None)

    Ls, Lb = splitBy( L.next, N)
    if L.item <= N:
        L.next = Ls
        Ls = L
     else:
        L.next = Lb
        Lb = L




    return (Ls, Lb)
```

***~~~ End of Paper ~~~***