FACULTY OF SCIENCE
ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	CSC03A3/CSC3A10: COMPUTER SCIENCE 3A
CAMPUS	AUCKLAND PARK CAMPUS (APK)
ASSESSMENT	SEMESTER TEST 1 2021 **MEMO**

DATE: 2021-03-25	**SESSION:** 14:00 - 16:00
ASSESOR(S):	MR R. MALULEKA PROF D.T. VAN DER HAAR
MODERATOR:	PROF D.A. COULTER
DURATION: 120 MINUTES	**MARKS:** 100

Please read the following instructions carefully:

1. You must complete this assignment yourself within the prescribed time limits.
2. You are bound by all university regulations please special note of those regarding assessment, plagiarism, and ethical conduct.
3. You must complete and submit the *"Honesty Declaration : Online Assessment"* document along with your submission to EVE. No submissions without an accompanying declaration will be marked.
4. Your answers together with the declaration must be submitted in a file named in the following format: STUDENTNUMBER_SURNAME_INITIALS_SUBJECTCODE_ASSESSMENT e.g. *202112345_SURNAME_IAM_CSC03A3_ST1.pdf*
5. Additional time for submission is allowed for as per the posted deadlines on EVE. If you are experiencing technical difficulties related to submission please contact us as soon as possible.
6. No communication concerning this test is permissible during the assessment session except with Academy staff members. The invigilator is available via email (dvanderhaar@uj.ac.za or rmaluleka@uj.ac.za) and on the "UJ CSC3A" Discord server throughout the assessment (https://discord.gg/76emAKMHZy).
7. This paper consists of 8 pages excluding the cover page.

QUESTION 1

(a) Consider the code below, which computes whether or not a given integer is a multiple of three. [10]

```
1  public boolean calc(int n){
2    return (n % 3 == 0);
3  }
```

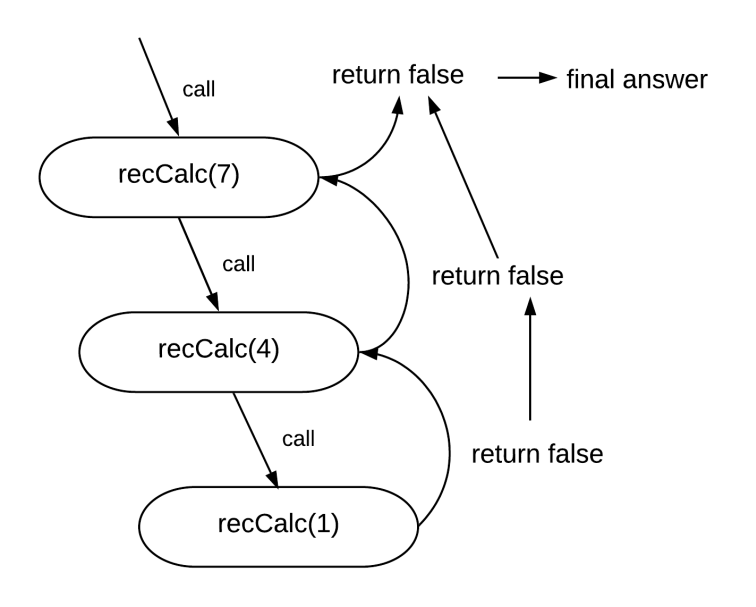
1. Create a recursive function that does the same task - recCalc. **Your function may not make use of division or the modulus operator**. (Hint: your method need only work for positive integers)
2. Draw a recursion trace for recCalc(7).

Solution:

1. 3 marks for base case, 2 marks for recursive call [5 marks]

```
1 public boolean recCalc(int n){
2   if (n<3) {
3     return (n == 0);
4   }
5   return recCalc(n-3);
6 }
```

2. 3 marks for correct call and return values, 2 marks for correct overall graphic [5 marks]



(b) Provide an appropriate Javadoc comment block for the function given in (a), calc. [4]

Solution:

- valid - opening and closing characters and asterisk on each line [1]
- description of function, e.g. Determines if an integer is divisible by three [1]

- @param n and a description, e.g. @param n an integer [1]
- @return and a description, e.g. @return true if n is divisible by 3; false otherwise [1]

(c) Imagine a function you have implemented is not working correctly. Discuss how you could **test** and **debug** your program. [3]

Solution:

max 3

- Run program on a representative set of inputs including special cases, e.g. base case [1]
- Generate (pseudo)random integers to test as inputs [1]
- Add print statements to track values of variables - possibly add temp variables [1]
- use IDE debugger to track variable values [1]
- replace with stub to test the rest of the program [1]

(d) Give the results of the following bit-wise operations (operands are binary numbers): [3]

1. 101 & 110
2. 101 ^ 110
3. 1010 >> 1

Solution:

1. 100 OR 4 [1]
2. 011 OR 3 [1]
3. 101 OR 5 [1]

Total: 20

QUESTION 2

(a) What is the asymptotic relationship between each of the following pairs of functions? [6]

1. n^3 and $n \log n$
2. π^n and n^e
3. $\log_4 n$ and $\log_2 n$

Solution:

1. n^3 is $\Omega(n \log n)$
2. π^n is $\Omega(n^e)$
3. $\log_4 n$ is $\Theta(\log_2 n)$

(b) Consider the following function and, using primitive counting, express the runtime of this function in Big-Oh notation. Let $n = arr.length$. [8]

```
1  public void selectionSort(int[] arr) {
2    int i, j, min, x;
3    for (i = 0; i < arr.length-1; i++) {
4      min = i;
5      for (j = i+1; j < arr.length; j++) {
6        if (arr[j] < arr[min])
7          min = j;
8      }
9      x = arr[min];
10     for (j = min; j > i; j--)
11       arr[j] = arr[j - 1];
12     arr[i] = x;
13   }
14 }
```

Solution:

```
1
2 4
3 1 + 4(n-1) = 4n-3
4 n-1
5 2(n-1) + 3(n-1)^2 = 3n^2 - 4n + 1
6 3(n-1)^2 = 3n^2 - 6n + 3
7 (n-1)^2 = n^2 - 2n + 1
8
9 2(n-1) = 2n - 2
10 (n-1) + 3(n-1)^2 = 3n^2 - 5n + 2
11 4(n-1)^2 = 4n^2 - 8n + 4
12 2(n-1) = 2n - 2
```

The result of primitive counting is $14n^2 - 16n + 7$. [6 marks (4 marks for different quadratic function)]
The runtime of the function is $O(n^2)$. [2 marks]

(c) Show that $2n^3 + n^2$ is $\Theta(n^3)$. [6]

Solution:

- $O(n^3)$: find c and n_0 such that $2n^3 + n^2 \leq cn^3$, for $n >= n_0$
  e.g. $c = 3, n_0 = 1$ [2]
- $\Omega(n^3)$: find c and n_0 such that $2n^3 + n^2 \geq cn^3$, for $n >= n_0$
  e.g. $c = 2, n_0 = 1$ [2]
- O and $\Omega \rightarrow \Theta$ [2]

Total: 20

QUESTION 3

(a) Stacks and Queues are two of the fundamental abstract data types that are often found in ADT libraries. Provide three points of comparison between a Stack and a Queue. [6]

Solution:
max 6

1. Insertions and removals: stack - LIFO; queue - FIFO [2]
2. Insertions and removals: both do not allows arbitrary inserts & removals [2]
3. Implementation: can both be implemented with an array or linked list with O(1) ops [2]
4. Fundamental operations: stack - push,pop; queue - enqueue, dequeue [2]
5. Java: stack - concrete implementation included; queue - only interface included [2]

(b) Consider the following list Interface and write a class *Deque*. Make use of the existing class List, which implements IList, and the Adapter design pattern to realize a *Deque ADT*. **Note: You do not need to implement the List methods.** In the interest of time, your Deque does not have to include the helper methods size and isEmpty. [14]

```
1  public interface IList<T> {
2      public Position<T> insertAfter(Position<T> p, T item);
3      public Position<T> insertFirst(T item);
4      public Position<T> insertLast(T item);
5      public T remove(Position<T> p);
6      public Position<T> first();
7      public Position<T> last();
8      public boolean isEmpty();
9      public Position<T> prev(Position<T> p);
10     public Position<T> next(Position<T> p);
11     public int size();
12 }
```

Solution:

Exceptions could be included but are not essential

```java
  //1 mark for formatting
public class Deque<T> {
  private List<T> listDeque; //2 marks

  public Deque() {
    listDeque = new List<T>(); //1 mark
  }

  //2 marks
  public void addFirst(T item) {
    listDeque.insertFirst(item);
  }

  //2 marks
  public void addLast(T item) {
    listDeque.insertLast(item);
  }

  //2 marks
  public T removeFirst() {
    Position<T> elem = listDeque.first();
    return listDeque.remove(elem);
  }

  //2 marks
  public T removeLast() {
    Position<T> p = listDeque.last();
    return listDeque.remove(p);
  }

  //1 mark
  public T getFirst() { //OR first()
    return listDeque.first().element();
  }

  //1 mark
  public T getLast() { //OR last()
    return listDeque.last().element();
  }
}
```

Total: 20

QUESTION 4

(a) Complete the skeleton class DequeIterator given below, that defines a **snapshot iterator** over a Deque using an underlying array. Provide **only the methods necessary** to make DequeIterator a valid **iterator**. NB, you may only make use of the standard Deque operations. [15]

```
public class DequeIterator<T> implements Iterator {
  private T[] arr;
  private int cursor;
  //Complete: Iterator class
    .
    .
    .
  @SuppressWarnings("unchecked")
  private T[] createArray(int size) {...}
    .
    .
    .
}
```

Solution:

```
public class DequeIterator<T> implements Iterator<T> {
  private T[] arr;
  private int cursor;

  //5 marks
  public DequeIterator(Deque<T> myDeque) {
    this.arr = createArray(myDeque.size());
    if (!myDeque.isEmpty()) {
      this.cursor = 0;

      for (int i=0; i<arr.length; i++) {
        arr[i] = myDeque.removeFirst();
        myDeque.addLast(arr[i]);
      }
    }
  }

  //5 marks
  @SuppressWarnings("unchecked")
  private T[] createArray(int size) {
    Object[] objArray = new Object[size];
    return (T[])objArray;
  }

  //2 marks
  public boolean hasNext() {
    return cursor<arr.length;
  }

  //3 marks
  public T next() {
    T element = arr[cursor];
    cursor++;
```

```
34        return element;
35    }
36 }
```

(b) We can create a growable Array List by replacing the underlying array with a larger one, whenever it is about to be full. Discuss the two **strategies** for creating a larger array, as well as their total and average insertion times. Which strategy has the better average runtime. [5]

Solution:

- Incremental strategy increases array length/capacity by constant c. [1 mark]
- Performance: $T(n)$ is $O(n^2)$. Average insert time is $O(n)$. [1 mark]
- Doubling strategy doubles array length/capacity when full. [1 mark]
- Performance: $T(n)$ is $O(n)$. Average insert time is $O(1)$. [1 mark]
- Doubling strategy has better average insertion time of $O(1)$. [1 mark]

Total: 20

QUESTION 5

(a) A binary tree can conveniently be stored in an array using the following function f to determine the index of the array at which to store a position p of the tree: [4]

- If p is the root of the tree, then $f(p) = 1$.
- If p is the left child of a position p, then $f(p) = 2f(p)$.
- If p is the right child of position q, then $f(p) = 2f(p) + 1$.

Use pseudocode to describe a **non-recursive** method for calculating the depth of a position p stored at index i in the array.

Solution:

Depth $\leftarrow$ floor($log_2 i$)

OR

count $\leftarrow$ 0
while $i > 1$
    $i \leftarrow i//2$
    count++

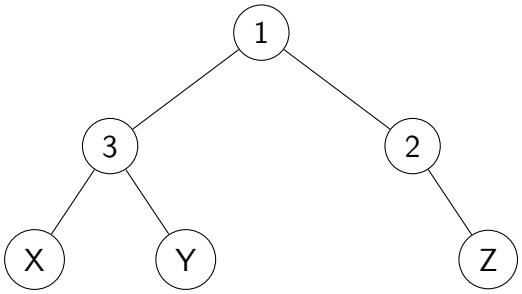
[1 mark for any recursive solution]

(b) What would the runtime of your depth algorithm be in big-oh notation? [2]

Solution:

$O(1)$ for 1st solution, $O(\log n)$ for 2nd, or whatever is correct for the answer for (a). [2 marks]

(c) Consider the tree below, and answer the questions that follow: [14]



1. List the elements in the order of a the following traversals:
   i. PreOrder traversal
   ii. PostOrder traversal
   iii. InOrder traversal
   iv. Euler tour traversal
2. What is the **height** of the given tree?
3. What is the **depth** of node with element 3?
4. Is the tree a proper binary tree? Why?

Solution:

1. 2 marks each for i.-iii. 4 marks for iv.
   i. 1 3 X Y 2 Z
   ii. X Y 3 Z 2 1
   iii. X 3 Y 1 2 Z
   iv. 1 3 XXX 3 YYY 3 1 22 ZZZ 2 1 [4 marks]
2. height $= 2$ [1 mark]
3. depth of $3 = 1$ [1 mark]
4. No, it is not a proper binary tree [1], because 2 is an internal node that does not have 2 children. [1]

Total: 20

— End of paper —