PAPUA NEW GUINEA UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

FIRST SEMESTER EXAMINATIONS - 2021

THIRD YEAR COMPUTER SCIENCE

CS314 – DATA STRUCTURES & ALGORITHMS

TIME ALLOWED: 3 HOURS

INFORMATION FOR CANDIDATES

1 Write your name and student number clearly on the front of the examination answer booklet.

2 You have 10 minutes to read this paper. You must not begin writing during this time.

3 There are TWO parts to this examination paper. You should attempt **ALL the questions.**

4 All answers must be written in the examination answer booklets provided.

5 Write the answers for question 1 to 30 on one page. Start the answer for each question beginning with question 31, on a **new** page. Do **not** use red ink or pencil.

6 Notes and textbooks are not allowed in the examination room.

7 Mobile phones and other recording devices are not allowed in the examination room.

8 Scientific and business calculators are not allowed in the examination room.

MARKING SCHEME

Marks are indicated at the beginning of each question. The total is **100** marks.

PART A - MULTIPLE CHOICES [1 x 30 = 30 MARKS]

For each of the questions in this section, you are required to select only ONE correct option and write the letter of that correct option on your answer booklet. All answers for this section should be placed on one page of the answer booklet.

Question 1
Which of the following (in Java) is more memory efficient considering the kinds of values to be stored?
(A) `int studentAge;` (C) `char studentAge;`
(B) `long studentAge;` (D) `byte studentAge;`

Question 2
Which of the following is a data structure?
(A) `int`. (C) Linked list.
(B) `String`. (D) Queue.

Question 3
What is the time complexity function (upper bound) of: $f(n) = 3n\log n + n^2 + 1500n + 60$
(A) O(n). (C) O(logn).
(B) $O(n^2)$. (D) O(nlogn).

Question 4
What is usually left out in (not covered by) the description of an Abstract Data Type (ADT)?
(A) The ADT's operations. (C) The ADT's implementation.
(B) The ADT's concept (what it is). (D) Both (A) and (C).

Question 5
Suppose that two functions (**X** and **Y**) are called sequentially to make up the body of a third function (**Z**). The two functions' time complexities are given as $g(n) = O(\log n)$ and $h(n) = O(n)$ respectively. What is ultimately the time complexity of **Z**?
(A) O(logn) + O(n). (C) O(n).
(B) O(nlogn). (D) O(logn).

Question 6
This is a collection of objects of the same type. Insertions and deletions can occur at any position in the collection. Ordering of the objects is important.
(A) Queue. (C) Stack.
(B) List. (D) Hashtable.

Question 7
Which ADT has the `front()` operation/method ?
(A) List. (C) Queue.
(B) Stack. (D) Tree.

Question 8
Which ADT is mostly best implemented using linked lists?
(A) List. (C) Queue.
(B) Stack. (D) Tree.

Question 9

What makes the `ArrayList` class different from the `LinkedList` class in Java?

(A) Queue ADT is only implemented using `ArrayList` and not `LinkedList`.
(B) `ArrayList` uses an array to store data objects while `LinkedList` does not.
(C) The methods of `ArrayList` work faster than their equivalent methods in the `LinkedList` class.
(D) None of the above is correct.

Question 10

Which statement correctly indicates that the class `Mimio` is a generic class?

(A) `class <T> Mimio {…}`
(B) `class Mimio <T> {…}`
(C) `<T> class Mimio {…}`
(D) `class <T> Mimio<T> {…}`

Question 11

In the generic class header you chose to be correct in the previous question (concerning class `Mimio`), is it possible to write a different label than `<T>` (for example `<TT>` or `<E>` or `<Truave>`, etc...) to define a generic type?

(A) Yes, it is possible.
(B) No, it is not possible.

Question 12

If you create a generic class, can you use it with a **primitive data type** in Java? (An example of this is shown with the generic class `Mimio` in the following code.)

```
Mimio<int> test = new Mimio<int>();
```

(A) Yes, it is possible.
(B) No, it is not possible.

Question 13

In a generic class, if you are going to make comparisons, you cannot use comparison operators such as <, >, or ==. Which of the following is a suitable solution?

(A) Redefine the comparison operators for the generic class.
(B) Find ways to avoid making comparisons in the generic class.
(C) Make the generic class extend the `Comparable` interface and use the `compareTo()` method.
(D) All of (A), (B), and (C) are equally suitable solutions.

Question 14

Suppose that a linked list contains nodes where one node at least has one link to another node in the list. There is a head pointer that points to the first node in the list. How do we determine if the linked list is empty?

(A) Check if the last node does not exist.
(B) Check if all the nodes on the linked list contains no data or nulls.
(C) Check if the head pointer is null.
(D) None of these is correct.

Question 15

Which of the following refers to the access approach in the Stack ADT?

(A) F-I-F-O
(B) L-I-F-O
(C) Both (A) and (B).

Question 16

What is the notion of a Priority Queue?

(A) A queue that is prioritized over other queues in a program.
(B) A queue with highly prioritized elements inserted first before less prioritized elements.
(C) A queue with highly prioritized elements removed first before less prioritized elements.
(D) Both (B) and (C) are correct.

Question 17

Which of the following is the **best** way to implement a priority queue?

(A) Using an array.
(B) Using a linked list.
(C) Using a heap.
(D) All of (A), (B), and (C) are equally suitable.

Question 18

A Tree data type contains nodes. What is another word for **leaf** node?

(A) Root.
(B) Edge.
(C) Depth.
(D) Terminal.

Question 19

Which tree traversal technique follows this order: `<left, root, right>` ?

(A) Inorder.
(B) Preorder.
(C) Postorder.

Question 20

BST is a type of tree, what does "BST" stands for?

(A) Breadth Search Tree.
(B) Binary Spanning Tree.
(C) Bubble Sort Tree.
(D) None of these is correct.

Question 21

Which of the following is true regarding a Tree?

(A) Trees are generally not good for applications where searching happens often.
(B) A node on a tree can have more than one parent node.
(C) The root node must always have two child nodes.
(D) A node that does not have any child node is called a leaf node.
(E) Both (C) and (D) are correct.

Question 22

Which of the following is an important characteristic of a Binary Tree?

(A) Each node should have at least two parent nodes.
(B) Each node should have at most two child nodes.
(C) A node should not have only one child node.
(D) Both (A) and (B).

Question 23

There are two common searching algorithms used with Graphs and Trees. They are abbreviated as DFS and BFS. What does BFS stands for?

(A) Binary Finite Search.
(B) Breadth First Search.
(C) Breakfast Search.
(D) Brick Force Search.

Question 24

In a hashtable, items are stored as key-value pairs. Which is hashed, the key or the value?

(A) Key.
(B) Value.
(C) Both (A) and (B).

Question 25

What is sufficient to know to access the items stored in a hashtable? (As a user, not implementer.)

(A) The keys for those items.
(B) The hashes of the keys for those items.
(C) The hashes of the values for those items.
(D) None of these is correct.

Question 26

Is the order of items in a hashtable important?

(A) Yes. (B) No.

Question 27

Can the keys of the items stored in a hashtable be Strings?

(A) Yes. (B) No.

Question 28

Which of these data types do you think is used in the print spooler program that manages print jobs on a network printer and allows documents to be printed in a first-come-first-serve manner accept that shorter jobs are pushed before longer jobs to be printed first?

(A) Stack. (C) Priority Queue.
(B) Hashtable. (D) Queue.

Question 29

An application stores the names of all the countries in the world together with their capital cities. What data type is best for this application to use to make searching of capital cities by country names faster/easier?

(A) Binary Search Tree. (C) Queue.
(B) Hashtable. (D) List.

Question 30

What data type is the best to use to store student names in a course group where I can easily insert or append names, sort the names in ascending or descending order, and remove any name I want?

(A) Binary Search Tree. (C) List.
(B) Hashtable. (D) Stack.

PART B - SHORT AND LONG ANSWERS [70 MARKS]

For this part, you are to begin your answer to each main question on a new page of the answer booklet. Write your answers clearly.

Question 31 [6 + 6 + 3 = 15 Marks]

This question is about Algorithm Analysis.

(A) Think about the algorithm for the problem given below.

Problem: Searching for a name in a list of ***n*** unsorted names.
Algorithm: Start with the first name and compare it with the searched name.
Stop if the names match.
Repeat this comparisons until the name is found or the end of the list is reached.

Briefly describe the best case, average case, and worst case scenarios of the algorithm. For each of these three cases, express the time complexity using Big-O.

(B) For the following algorithm/method, show both the time (`T(n)`) and space (`S(n)`)functions as well as the complexity functions (expressed using Big-O). Write brief notes in your working to show your reasoning.

```
void test(int A){ //A is an array.
    int n = A.length;
    for(int i = 1; i <= n; i = i+i)  //Consider also the statements involving i.
           { System.out.print("Hello"); }
    }
}
```

(C) Showing working, find the upper bound for this time function: $f(n) = n^2 + 4n$.

Question 32 [4 + 4 + 3 + 2 + 2 = 15 Marks]

This question is about Algorithm Analysis to be done on an ADT's implementations using array and linked lists. Provided in Appendix A are two Stack implementations (array-based and linked-list-based). You are to study them when answering the following questions (A) to (C).

(A) Write the time complexity functions for the following operations in the array-based Stack:
`push(), pop(), peek(), size()`.
(B) Write the time complexity functions for the following operations in the linked list-based Stack:
`push(), pop(), peek(), size()`.
(C) State how you can improve the speed of the **`size()`** operation in the linked list-based Stack to **`O(1)`**.

The two question parts below are about tracing a stack and a queue.

(D) Follow the code for a stack given below and state what the value of `remVal` will be at the end.

```
Stack<Integer> pile = new Stack<Integer>();
pile.push(160);
pile.push(921);
pile.push(41);
int remVal = pile.pop();
pile.push(10);
remVal = pile.peek();
pile.pop();
remVal = pile.pop(); //(What is remVal's value finally?)
```

(E) Follow the code for a queue given below and state what the value of `name` will be at the end.

```
Queue<String> line = new Queue<String>();
line.enqueue("Luther");
line.enqueue("Greg");
String name = line.dequeue();
line.enqueue("Maria");
line.enqueue("Linda");
name = line.peek();
line.enqueue("Thomas");
name = line.dequeue();
name = line.dequeue(); //(What is name's value finally?)
```

Question 33 [5 + (2 + 2 + 2) = 11 Marks]

This question is about Binary Search Trees and Tree ADT in general.

(A) Take the name NICKOPARALOZE. You are to construct (draw) a Binary Search Tree containing the characters of this name. First begin by specifying the first letter of this name to be the root node.

(B) List the characters on the Binary Search Tree you created in (a) (comma separated list) the way you traverse them using:

- (i) Inorder traversal
- (ii) Preorder traversal
- (iii) Postorder traversal

Question 34 [6 + 2 + 4 = 12 Marks]

This question is about Hashtable and Priority Queue.

(A) A hashtable can experience collision with hashes where some items have the same hash. State and explain the two ways to resolve hash collisions.

(B) In one or two sentences briefly differentiate between the Queue and Priority Queue ADTs.

(C) Comment on (or describe) the insertion and deletion operations of the priority queue.

Question 35 **[2 + 3 + 6 + 3 + 3 = 17 Marks]**

Refer to Appendix B for the code of a simple List class. This List only store int values. Study the code and complete the codes for the following methods whose method signatures and comments are shown in Appendix B. Write out the full method code for each of them.

(A) `get()`
(B) `replace()`
(C) `append()`
(D) `size()`
(E) `isEmpty()`

END OF EXAMINATION

APPENDIX A

Stack –Array-based implementation, only storing Strings.

```
public class TestStack {
   //Has operations push(), pop(), peek(), size(), isEmpty().
   String data[];
   BSCStack(){ } //Can also leave this line out to be implied since it's the only constructor.

   public void push(String newItem) {
     if (isEmpty()) {
       //If data array is empty then...
       data = new String[1];
       data[0] = newItem;
     }
     else {
       //If data array already has some elements then...
       String[] temp = new String[data.length + 1];
       //Copy data items into temp array.
       System.arraycopy(data, 0, temp, 0, data.length);
       //Put new item in last cell of temp array.
       temp[data.length] = newItem;
       //Point data to temp array.
       data = temp;
       temp = null; //For garbage collection. Java can handle that automatically so we can drop
            //this line of code.
       }
       }//End of push() method.

   public String pop() {
       //The recent element added is to be removed. That means removing the item in the last
       //cell of data array.
       if (isEmpty()) {
              //If data array is empty then...
              return ""; //Return a blank string.
       }
       else {
              String[] temp = new String[data.length-1];
              //Copy all items in data array except the last item into temp array.
              System.arraycopy(data, 0, temp, 0, data.length-1);
              //Keep a copy of the item in last cell of data array.
              String lastItem = data[data.length-1];
              //Set data array to the temp array.
              data = temp;
              //Destroy temp array (optional).
              temp = null;
              //return the last item.
              return lastItem;
       }
   }//End of pop() method.

   public String peek() {
       //Return the last item in data array if data array is not empty.
       if (isEmpty()) {
```

```
            return ""; //A blank string.
        }
        else {
            return data[data.length];
        }
    }

    public int size() {
        //Returns the count of all the items in data array.
        return data.length;
    }

    public boolean isEmpty() {
        return (data==null);
    }

    public String toString() {
        String items = "";
        for (String i: data) {
            items += i + " ";
        }
        items += "<--Top";
        return items;
    }
}
```

Stack ---Linkedlist-based implementation, only storing Strings.

```
public class TestStack {
    //Has operations: push(), pop(), peek(), size(), isEmpty().
    Item top;

    //I define an Item class to create Item objects that will be stored
    //on a stack.
    class Item{
        int data;
        Item down; //down is a reference to the next Item below the current one on the stack.
        Item (int data){
            this.data = data;
        }
    }

    public boolean isEmpty() {
        return top == null;
    }

    public void push(int value) {
        //Create new Item object.
        Item newItem = new Item(value);
        //Add new item to the top of the stack.
        //If stack is empty, set new item to top.
        if (isEmpty()) {
            top = newItem;
```

```
            return;
        }
        //Else if stack is not empty, set newItem as top.
        Item temp = top;
        top = newItem;
        top.down = temp; //Could also have: newItem.down = temp;
    }

    public int size() {
        //Run through all the Item objects on the stack and count them.
        Item temp = top;
        int count = 0;
        while (temp != null) {
            temp = temp.down;
            count++;
        }
        return count;
    }

    public int peek() {
        return top.data;
    }

    public int pop() throws ArrayIndexOutOfBoundsException{
        if (size()==0) throw new ArrayIndexOutOfBoundsException("Nogat item long stack.");
        //If size() >= 1, then below code will run.
        Item temp = top.down;
        int d = top.data;
        top = temp;
        return d;
    }

    //Notice that this toString() method enables me to supply the name of MyStack objects
    //into the System.out.print() function in my application code and get a nice display.
    public String toString() {
        String msg = "";
        Item temp = top;
        while (temp != null) {
            msg += "|" + temp.data + "\n";
            temp = temp.down;
        }
        return msg;
    }
}
```

APPENDIX B

```
public class MyList {
  //Data
  private int[] data;

  //Operations/Methods

  public void insert(int index, int value){
            int[] newArr = new int[data.length + 1];
            System.arraycopy(data, 0, newArr, 0, index);
            System.arraycopy(data, index, newArr, index + 1, data.length - index);
            newArr[index] = value;
            //Reference data to newArr.
            data = newArr;
  }

  public int get(int index) {
  //Returns the value at the cell index in data array.

  }

  public void replace(int index, int value) {
  //Replaces the current value at cell index in data array with the new value provided
  //here in the parameter list.

  }

  public void append(int value) {
  //Adds the value at the end of the data array.

  }


  public int size() {
  //Returns the total number of int values in data array.

  }

  public boolean isEmpty() {
  //Returns true if data array is empty or false otherwise.

  }

}//End of class code.
```