Singapore Polytechnic

School of Electrical & Electronic Engineering

ET0736 Object Oriented Programming and Data Structure

2024/2025 Semester 1 Lab Test 3 Sample

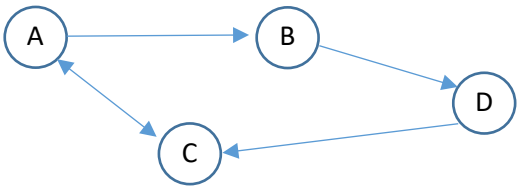
TOTAL MARKS

/100

1. Below is a simple class **Vertex** to represent a node or vertex in a graph data structure. The **ArrayList adjList** stores the list of vertices which are accessible from the vertex.

```
class Vertex{
    String label;
    ArrayList<Vertex> adjList = new ArrayList<>();
    Vertex (String label) {   this.label = label;  }
}
```

The program segment below intends to construct a Java **Graph** structure for this route given below using the class **Vertex** provided. The **adjList** for Vertex object **a** has been initialised. Complete the initialisation of **adjList** for the **b**, **c** and **d Vertex** objects. [**20 marks**]



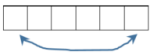
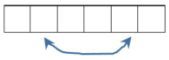

```
Vertex a = new Vertex("A");
Vertex b = new Vertex("B");
Vertex c = new Vertex("C");
Vertex d = new Vertex("D");

a.adjList.add(b);
a.adjList.add(c);  // write your answers in space below
```

2. The recursive method ***isPalindrome()*** is to check if a given string is a palindrome. The **main()** method is good. Do not change it except the value of the string **a** for testing purposes. [**30 marks**]

Palindrome : *a word that reads the same backwards as forwards.*
(Examples: level, radar, pop, civic etc)

Intended algorithm of ***isPalindrome()***:

- Compare characters in [0] and [length-1] 
  - If different, stop the recursion and return False (i.e. given string is not a palindrome)
  - If same, repeat comparison for characters in [1] and [length-2] 
- When recursion hits the middle of the string and still same characters, return TRUE.

Debug the ***isPalindrome()*** below:

```
public static void main(String[] args) {
   String a = new String("abcdcba"));
   if (isPalindrome(a, 0, a.length()-1))
     System.out.println(a + " is a palindrome");
   else
     System.out.println (a + " is NOT a palindrome");
}

public static void isPalindrome (String x, int i, int j ){
    if (i<(x.length()-1) {
       if (x.charAt(i)!=x.charAt(j)) return false;
       else {
          isPalindrome(x,i,j);
       }
    }
    return (true);
}
```

Write the corrected ***isPalindrome()*** below:

```
public static boolean isPalindrome (String x, int i, int j ){

}
```

3. Below is the code for a simple custom linked list class ***MyLinkedList***. **[50 marks]**

```
class MyLinkedList<E> {
    Node<E> head = null;
    Node<E> current = null;
    Node<E> newNode;
    public void append (E x){
       newNode = new Node<E> (x);

       if (head==null) {
         // for very first mode
         head = new Node(x);
       }
       else {
          current = head;
          // track down to tail node
          while (current.next != null)
             current = current.next;
          // add in the new node
          current.next = newNode;
     }
   }

     public String toString (){
          String s=“”;
          current = head;
          while {
              s += current.data.toString();
              current = current.next;
          }
          return(s);
      }
}

class Node<E> {
    E data;
    Node<E> next = null;
    Node (E data) { this.data = data; }
}
```

Add a method **size()** to return the number of elements in the linked list.

```
int size()
```

Add a method **remove()** to remove an element from the linked list at the index.
Check if the index is valid. Return -1 if index is invalid. Return 0 if deletion is successful.

```
int remove(index)
```