UNIVERSITI TUNKU ABDUL RAHMAN

ACADEMIC YEAR 2019/2020

QUIZ

UECS2083/UECS2413 PROBLEM SOLVING WITH DATA STRUCTURE AND ALGORITHMS

SATURDAY, 10TH AUGUST 2019 TIME :1.00 PM – 1.40 PM (40 minutes)

BACHELOR OF ENGINEERING (HONOURS) ELECTRICAL AND ELECTRONIC ENGINEERING
BACHELOR OF SCIENCE (HONS) APPLIED MATHEMATICS WITH COMPUTING
BACHELOR OF SCIENCE (HONS) SOFTWARE ENGINEERING

Instruction to Candidates:

This question paper consists of 3 questions.

Answer all the questions in this question paper.

Student ID : ____

Name : ____

Programme : ____

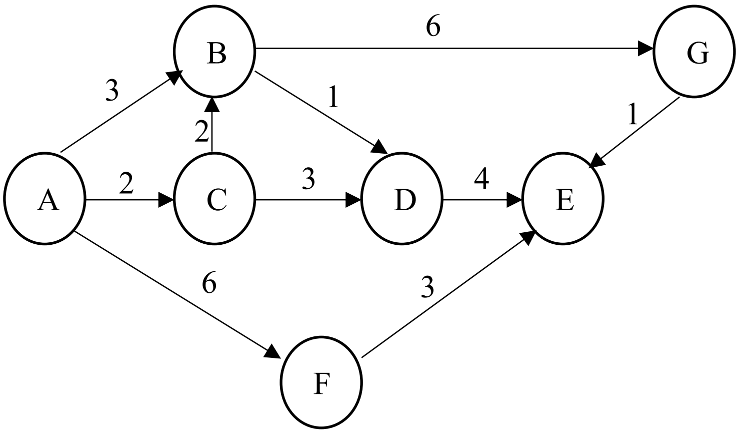
Lecture Group : ____

	MARKS
Question 1	
Question 2	
Question 3	
TOTAL:	

Question 1

Consider the following weighted graph.

***Assume that the adjacency follows the order of vertex representative alphabet sequence.*



(a) Construct an ***adjacency matrix*** representation for this graph. (10 marks)

[Answer:]

	A	**B**	**C**	**D**	**E**	**F**	**G**
A	**0**	**3**	**2**	**0**	**0**	**6**	**0**
B	**0**	**0**	**0**	**1**	**0**	**0**	**6**
C	**0**	**2**	**0**	**3**	**0**	**0**	**0**
D	**0**	**0**	**0**	**0**	**4**	**0**	**0**
E	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F	**0**	**0**	**0**	**0**	**3**	**0**	**0**
G	**0**	**0**	**0**	**0**	**1**	**0**	**0**

(b) Starting at vertex A, trace a ***breadth-first traversal*** through the above graph. You are required to show the ***queue*** contents as you work your way down the graph. (10 marks)

[Answer:] Mark allocation: For the correct BFS output and the queue content

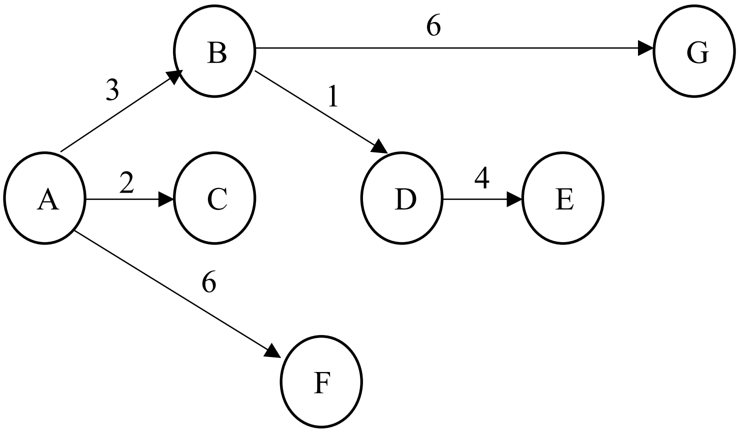
OUTPUT: A, B, C, F, D, G, E

Steps:	**Queue Content**
1. Start at vertex A, enqueue A to Queue and mark as visited	A
2. While queue not empty: Dequeue an element: A A's unvisited adjacent nodes = B, C, F - Enqueue B, mark as visited - Enqueue C, mark as visited - Enqueue F, mark as visited	B | C | F
3. While queue not empty: Dequeue an element: B B's unvisited adjacent nodes = D and G - Enqueue D, mark as visited - Enqueue G, mark as visited	C | F | D | G
4. While queue not empty: Dequeue an element: C C's unvisited adjacent nodes = none	F | D | G
5. While queue not empty: Dequeue an element: F F's unvisited adjacent nodes = E - Enqueue E, mark as visited	D | G | E
6. While queue not empty: Dequeue an element: D D's unvisited adjacent nodes = none	G | E
7. While queue not empty: Dequeue an element: G G's unvisited adjacent nodes = none	E
8. While queue not empty: Dequeue an element: E E's unvisited adjacent nodes = none	
9. Queue is empty, stop searching	

(c) Suppose the source vertex is A, develop a ***shortest path*** for the graph using Dijkstra's algorithm. (10 marks)

[Answer:]

	A	B	C	D	E	F	G
Cost	0	3	2	4	8	6	9
Parent	-1	A	A	B	D	A	B



Step 1: Start at A
A → B (3)
A → C (2) **
A → F (6)

Step 2: Connected vertices, A and C
A → B (3) ** A → C → B (2+2 =4)
A → F (6) A → C → D (2+3=5)

Step 3: Connected vertices, A, B, C
A → F (6) A → C → D (2+3=5) A → B → D (3+1= 4) **
A → B → G (3+6=9)

Step 4: Connected vertices, A, B, C, D
A → F (6)** A → B → G (3+6=9) A → B → D → E (3+1+4=8)

Step 5: Connected vertices, A, B, C, D, F
A → B → G (3+6=9) A → B → D → E (3+1+4=8) ** A→ F → E (6+3=9)

Step 6: Connected vertices, A, B, C, D, E, F
A → B → G (3+6=9)

Step 6: Connected vertices, A, B, C, D, E, F, G; All vertices now connected.

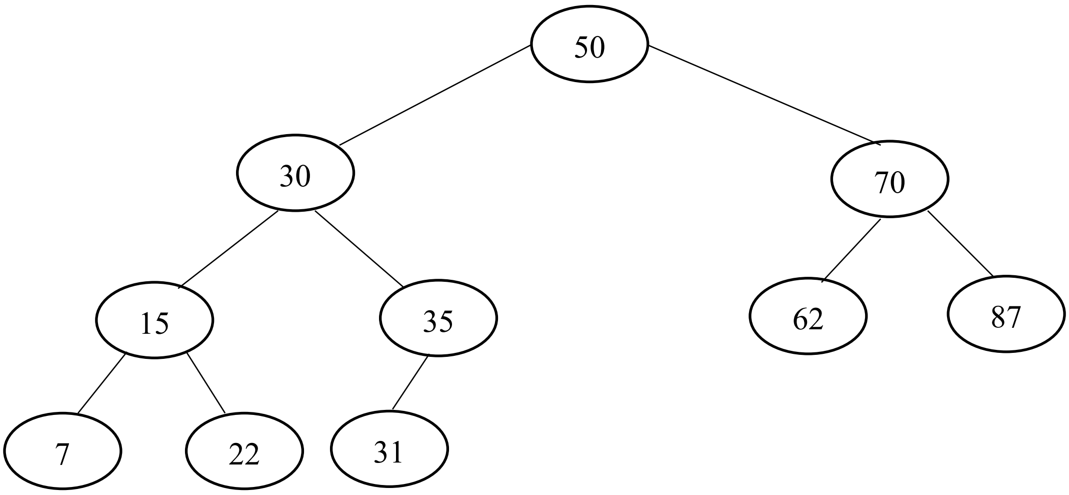
Question 2

A binary search tree has 10 nodes whose data field are integers. The ***inorder*** and ***postorder*** traversals of the tree are given below. Draw the binary tree showing the data in each node and the references between the nodes.

In-order	:	7	15	22	30	31	35	50	62	70	87
Post-order	:	7	22	15	31	35	30	62	87	70	50

(10 marks)

[Answer:]

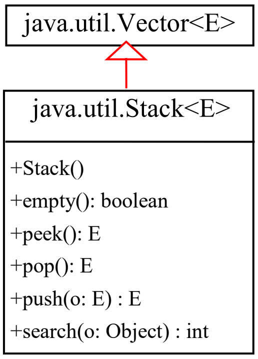


Question 3

Suppose that there is an empty stack. Write a program that ***print the content*** of the stack after performing each of the following operation:

- Insert “Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday” and “Sunday” onto the stack
- Remove two elements from stack
- Return the top element
- Check if the stack is empty
- Check if “Monday” exist in the stack

Refer to the Stack class given below.



Creates an empty stack.
Returns true if this stack is empty.
Returns the top element in this stack.
Returns and removes the top element in this stack.
Adds a new element to the top of this stack.
Returns the position of the specified element in this stack.

(10 marks)

[Answer:]

```
import java.util.Stack;

public class StackForQuiz {

  public static void main(String[] args) {
     Stack<String> stack = new Stack<String>();                [1]

     stack.push("Monday");
     stack.push("Tuesday");
     stack.push("Wednesday");
     stack.push("Thursday");                                   [2]
     stack.push("Friday");
     stack.push("Saturday");
     stack.push("Sunday");
     System.out.println(stack);
     stack.pop();
     stack.pop();                                              [1]
     System.out.println(stack);
     System.out.println("First element is :" +
                                             stack.peek());    [2]
     System.out.println("Is the stack empty?" +
                                          stack.isEmpty());    [2]
     System.out.println(stack.search("Monday"));               [2]

     }
}
```