| Index Number: | | | | | | | |
|---|---|---|---|---|---|---|---|

# UNIVERSITY OF MORATUWA

## FACULTY OF ENGINEERING

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BSc Engineering Honours Degree
2013 Intake Semester 2 Examination

# CS2022: DATA STRUCTURES & ALGORITHMS

Time allowed: 2 Hours March 2015

**ADDITIONAL MATERIAL:** *None*

**INSTRUCTIONS TO CANDIDATES:**

1. This paper consists of **two (2) Sections** in **14** pages.
2. Answer **all** questions in the question paper itself **in the space provided**. If you make a mistake or need additional space you may attach additional sheets.
3. Section **A** contains **twenty-five (25) MCQ and/or short answer questions**. **Answer these questions in the space provided in the paper**.
4. **For each MCQ question** in section A, there is **only one correct answer** and you are expected to **clearly mark only one choice**.
5. Each question in section A carries 2 marks.
6. Section B contains **two (2)** essay type questions.
7. The maximum attainable mark for each question in section B is given in brackets.
8. The maximum attainable mark for this paper is 100. This examination accounts for 60% of the module assessment.
9. This is a closed book examination.

   ***NB: It is an offence to be in possession of unauthorized material during the examination.***

10. Only calculators approved and labelled by the Faculty of Engineering are permitted.
11. Assume reasonable values for any data not given in or with the examination paper. Clearly state such assumptions made on the script.
12. In case of any doubt as to the interpretation of the wording of a question, make suitable assumptions and clearly state them on the script.
13. This paper should be answered only in English.

## SECTION A

1. Which of the following factors are important in comparing two algorithms?

   I. Efficiency of the algorithms
   II. Space required for execution of the algorithms
   III. Simplicity of the algorithms

   a) I Only
   b) I and II only
   c) I and III only
   d) All three

2. Which of the following statements are true regarding algorithms?

   I. An algorithm is a well-defined procedure that will take some inputs and produce some outputs.
   II. A deterministic algorithm may give different answers in different executions for the same input.
   III. Recursive algorithms cannot be expressed using flow charts.

   a) I Only
   b) I and II only
   c) I and III only
   d) All three

3. Following array is to be sorted using insertion sort.

| 25 | 12 | 47 | 33 | 46 | 55 | 35 | 70 |
|---|---|---|---|---|---|---|---|

   The array after first and second iteration of insertion sort are shown below.

   After First Iteration:

| 12 | 25 | 47 | 33 | 46 | 55 | 35 | 70 |
|---|---|---|---|---|---|---|---|

   After Second Iteration:

| 12 | 25 | 47 | 33 | 46 | 55 | 35 | 70 |
|---|---|---|---|---|---|---|---|

   What will be the array after the **forth** Iteration of the Insertion sort?

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

4. Which of the following statements are true regarding bubble sort?

   I. Bubble sort sorts in place.
   II. If the array is sorted, there will be no swaps in a single iteration of the bubble sort.
   III. After *k* iterations of the bubble sort, the first *k* elements of the array is guaranteed to be in the sorted order.

   a) I Only
   b) II only
   c) I and II only
   d) II and III Only

*Continued …*

5. Which of the following statements are true regarding the efficiency of sorting algorithms?

    I. Worst case time complexity of heap sort is $O(n\log n)$.
    II. Quick sort is a more efficient soring algorithm than insertion sort.
    III. Worst case time complexity of bubble sort and quick sort are the same.

    a) I Only
    b) I and II only
    c) I and III only
    d) All three

6. If $f(x) = 8x^2 + 7x + 25$ and $g(x) = x^3 + 2x^2 + 4$ and $h(x) = 25x^2 + 25$ which of the following statements is **wrong**?

    a) $f(x) \in o(g(x))$ and $g(x) \in \omega(h(x))$ and $h(x) \in o(f(x))$
    b) $f(x) \in o(g(x))$ and $f(x) \in O(g(x))$ and $f(x) \in o(h(x))$
    c) $g(x) \in \Omega(h(x))$ and $g(x) \in \omega(h(x))$ and $g(x) \in \Omega(f(x))$
    d) $h(x) \in \Theta(f(x))$ and $h(x) \in \Omega(f(x))$ and $h(x) \in O(g(x))$

7. Suppose a divide and conquer algorithm solves a problem by dividing the problem into *m* sub-problems and then solving k sub-problems out of it. The recursive function that represents the time complexity of the algorithm, $T(n)$ could be;

    a) $T(n) = T\left(\frac{n}{k}\right) + f(n)$
    b) $T(n) = kT\left(\frac{n}{m}\right) + f(n)$
    c) $T(n) = mT\left(\frac{n}{k}\right) + f(n)$
    d) $T(n) = mT\left(\frac{n}{2}\right) + f(n)$

8. Which of the following is not a divide and conquer algorithm?

    a) Binary search
    b) Heap sort
    c) Merge sort
    d) Quick sort

9. What is an abstract data type (ADT)?

    ……..

    ……..

10. Which of the following statement is true regarding the data structures?

    I. Arrays can hold only homogeneous data.
    II. Records are used to hold heterogeneous data relevant to a single entity.
    III. Arrays, lists and stacks are linear data structures.

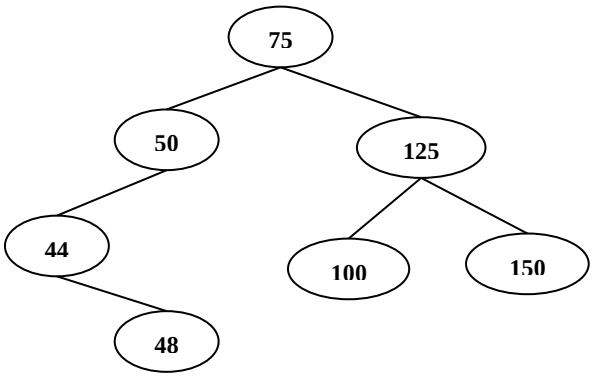
    a) I Only
    b) I and II only
    c) I and III only
    d) All three

*Continued …*

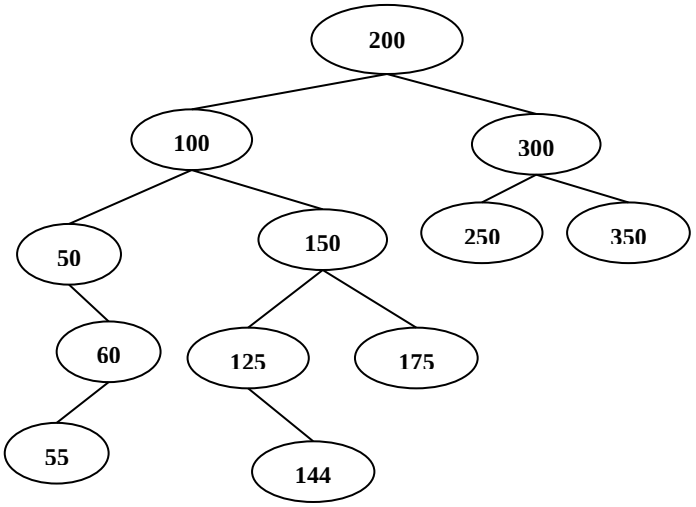
11. The in-order traversal of a balanced binary tree resulted in A, B, C, D, E, F, G. What will be the pre-order traversal result of the same tree?

……..

12. Extend the following binary search tree by adding 45 and 55 to it.




13. Show the changes that will happen to the following binary search tree if 100 is deleted from the tree.




*Continued …*

***Use the following description to answer questions 14, 15 and 16.***

*You are given a partial heap implementation with the following methods.*

- `int HeapSize(A)`

  Will return the actual number of elements in the array (heap).

- `void Max-Heapify(Array A, int i)`

  Given a node ***i***, with left child ***l*** and a right child ***r*** and with both the sub -trees rooted by ***l*** and ***r*** are max-heaps, this function will convert the sub-tree rooted by node ***i*** into a max-heap.

- `void Max-HeapInsertElement(Array A)`

  This function will insert a dummy element with value 0 as the last element of the heap.

- `void HeapIncreaseKey(Array A, int i, int amount)`

  This function will increase the key value of the node ***i*** in the heap by the amount specified by amount and move the $i^{th}$ node to the proper location in the heap.

14. The BuildMax-Heap operation (`void BuildMax-Heap(Array A)`) which will convert the specified array into a max-heap needs to be implemented. Complete the pseudocode for BuildMax-Heap operation shown below.

```
void BuildMax-Heap(Array A){
     heap_size = HeapSize(A)

     for (i= …… down to 1

          ……

}
```

15. The Max-HeapInsert operation (`void Max-HeapInsert(Array A, int val)`) which will insert a new element with the specified value (specified by val) to the heap needs to be implemented. Complete the pseudocode for Max-HeapInsert operation shown below.

```
void Max-HeapInsert(Array A, int val){
     Max-HeapInsertElement(A)
     heap_size = HeapSize(A)

     ……

}
```

*Continued …*

16. The HeapExtractMax operation (`int HeapExtractMax(Array A)`) which will remove the maximum value from the heap and return it needs to be implemented. Complete the pseudocode for HeapExtractMax operation shown below.

```
int HeapExtractMax(Array A){
     heap_size = HeapSize(A)
     max = A[0]

     ……

     ……

     return max
}
```

17. Open addressing and chaining are two methods of collision handling in hash tables. **Draw a suitable diagram** to illustrate the structure of a hash table which uses chaining for the collision handling.

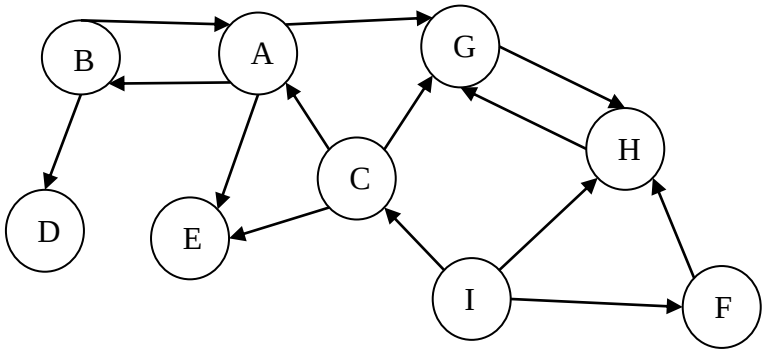
18. Which of the following statements are correct regarding the graph representations?

    I. **Adjacency matrix** representation allows **fast checking the presence of an edge**.
    II. If the graph is a **sparse** graph with **large number of vertices**, then **adjacency list** representation is more **space efficient**.
    III. If the adjacency matrix of a graph is **symmetric**, it **must** be an **undirected** graph.

    a) I Only
    b) I and II only
    c) I and III only
    d) All three

*Continued …*

***Use the following graph to answer questions 19, 20 and 21.***




19. A possible breadth first search (BFS) result starting from vertex A can be shown as A, ( B, E, G), (D, H), **in which the B, E and G are indicated within brackets to indicate that the order among them can vary according to the selection order of vertices**. Similarly D and H are within a different pair of brackets to indicate their order can change. Which of the following is **not** a valid BFS result for the above graph?

    a) I, (C, H, F), (A, E, G), (B, D)
    b) C, (A, E, G), (B, H), D
    c) F, H, G
    d) All of the above are valid BFS results

20. Which of the following are **valid** discover and finish time stamps (discover, finish) for vertices A and E respectively for some depth first searches (DFS)?

    I. A: (3, 10) and E: (4, 5)
    II. A: (8, 13) and E: (2, 3)
    III. A: (2, 7) and E: (8, 9)

    a) I Only
    b) I and II only
    c) I and III only
    d) II and III only

21. Which of the following statement is **not correct** regarding the results of depth first search (DFS) of the above graph?

    a) If DFS starts from vertex A, edges BA and HG will be back edges.
    b) If DFS starts from vertex I, edges CG and IH could become forward edges.
    c) If DFS starts from vertex I, there will be no cross edges.
    d) All three of the above statements are correct.

22. Which of the following is **not** a greedy algorithm?

    a) Bellman-Ford algorithm for generating single source shortest path.
    b) Dijkstra's algorithm for generating single source shortest path.
    c) Kruskal's minimum spanning tree generation algorithm.
    d) Prim's minimum spanning tree generation algorithm.

*Continued …*

23. What is optimal substructure property?

……

……

……

……

24. What is greedy-choice property?

……

……

……

……

25. Which of the following statements are **not** correct regarding the space complexity of sorting algorithms?

    a) Space complexity of heap sort and merge sort are the same.
    b) Space complexity of quick sort and heap sort are the same.
    c) Space complexity of insertion sort and bubble sort are the same.
    d) Space complexity of bubble sort and heap sort are the same.

*Continued …*

## SECTION B

**Q1.** **[25 *marks*]**

(a) You are given a list implementation, which holds a list of integers with the following functions.

- `int LIST-HEAD(list L)`: Returns the first element (an integer) in the list. **If the list is an empty list, this will cause the program to crash** (it assumes that the list has at least one element).
- `list LIST-REST(list L)`: Returns a new list with the first element of the original list removed. If the original list contains only one element, this will return an empty list.

Write the pseudocode of find_max function, which will accept a list (from above implementation) as the parameter and return the largest number of the input list. You may assume you have a function MAX (`MAX(int a, int b)`), which accepts two integers and returns the larger value of the two. [8 *marks*]

```
int FIND_MAX(list L){
        ……
        ……
        ……
        ……
        ……
        ……
        ……
        ……
}
```

(b) Pseudocode for merge sort is shown below.

```
MERGE-SORT(A, p, r)
1.  IF p < r
2.      q ← ⌊(p + r)/2⌋
3.      MERGE-SORT(A, p, q)
4.      MERGE-SORT(A, q + 1, r)
5.      MERGE(A, p, q, r)
```

*Continued …*

The merge algorithm has a worst case time complexity of $\theta(n)$. Show that the worst case time complexity of merge sort is $\theta(n\log n)$ [9 *marks*]

*Continued …*

(c) You are asked to implement two stacks in one array A[1…n] in such a way that neither stack overflows unless the total number of elements in both stacks together is n. Explain, using pseudo-code how PUSH ( Stack, x), POP (Stack) operations are implemented in the given scenario. (The PUSH and POP operations should run in $O(1)$ time.) [8 *marks*]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

*Continued …*

**Q2.** **[25 *marks*]**

(a) Identify the most efficient data structure for each of the following scenarios and justify your answer in one or two sentences [2 *marks* x 4]

i) To maintain a list of students where the student identification key comes from a large key space. Also, it has been established that it is impossible to come up with a good hashing function for the keys. The records will be searched very frequently, but changes are very infrequent.

……

……

……

ii) Build a keyword index for an e-book. Search for the list of pages corresponding to a keyword should happen in nearly O(1) time.

……

……

……

iii) To implement an algorithm to reverse any given string.

……

……

……

iv) Going through a large text file and counting the number of times each word appearing in the text.

……

……

……

(b) You are required to implement the road map of the Colombo area to identify the fastest routes. The city limit has road segments which connect the intersections. Certain road segments are bidirectional and some others are unidirectional. A graph will be a suitable data structure to represent this scenario. Explain how following will be represented in the graph. [1 *marks* x 4]

i) Intersections: ……..……

ii) Road segments ……..

*Continued …*

iii) Time to travel between road segments: ……

……

iv) Direction of traffic flow in road segments: ……

……

(c) In addition to the information in part (b), you have observed that the police keeps changing the traffic flow directions of the roads very frequently. Considering all these, **explain** why adjacency matrix representation is a suitable approach for implementing graph that represents the road network. [2 *marks*]

……

……

……

……

(d) An array of integers is shown below. Heap sort will be used to sort the array.

i. Show how the BUILD-MAX-HEAP(A) will convert the array into a heap. You are required to indicate each function call (each call to Max-Heapify) and the resultant array after that function call returns. [5 *marks*]

| **Function Call** | **Resultant Array** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Index | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Initial Array | **23** | **35** | **50** | **60** | **80** | **90** | **100** | **75** |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

*Continued …*

ii. Show how the resultant array of part (i) will be sorted by the HEAP-SORT. You are expected to indicate each change to the heap (reducing the size of the heap, changing of an element in the heap, calling Max-Heapify). You have to only show how the three largest numbers are moved to the three last positions of the array. Showing of complete sorting process is not required. [6 *marks*]

| **Function Call** | **Resultant Array** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Index | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Initial Heap | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

--- ***End of the Paper*** ---