**Question 1 (Mandatory) (1 point)**

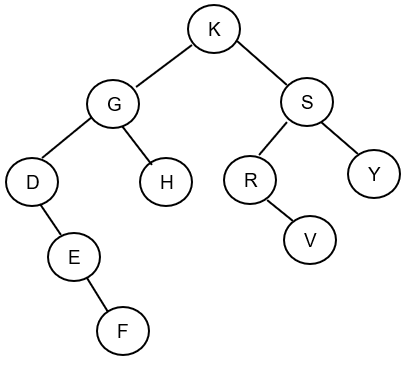
The best hash functions are characterized by:

Question 1 options:

|  |  |
| --- | --- |
|  | The **hash function** selection requires knowledge of key values and their distribution. |
|  | The **hash function** "uniformly" distributes the data across the entire set of possible **hash** values |
|  | The **hash function**  has **two-way** hashing algorithm, or in other words, the hash can be converted back into the original key. |
|  | The calculated **hash** code is fully determined by the data being hashed. |

**Question 2 (Mandatory) (1 point)**

Here is a binary tree:



Write the node labels in the order they would be printed in an in-order traversal of the tree.

Question 2 options:

|  |
| --- |
|  |
| V 9 |
|  |

|  |
| --- |
| G 4 |
|  |

|  |
| --- |
| H 5 |
|  |

|  |
| --- |
| F 3 |
|  |

|  |
| --- |
| S 8 |
|  |

|  |
| --- |
| D 1 |
|  |

|  |
| --- |
| Y 10 |
|  |

|  |
| --- |
| R 7 |
|  |

|  |
| --- |
| K 6 |
|  |

|  |
| --- |
| E 2 |

**Question 3 (Mandatory) (1 point)**

In a linked list data structure, when does the reference to the first node need to be updated?

I inserting into an empty list

II deleting from a list with one node

III deleting second node

Question 3 options:

|  |  |
| --- | --- |
|  | III |
|  | I |
|  | I and II |
|  | II |

**Question 4 (Mandatory) (1 point)**

How many references need to be updated when we remove a node from the end of a doubly linked list?

Question 4 options:

|  |  |
| --- | --- |
|  | 2 |
|  | 1 |
|  | 3 |
|  | 4 |

**Question 5 (Mandatory) (1 point)**

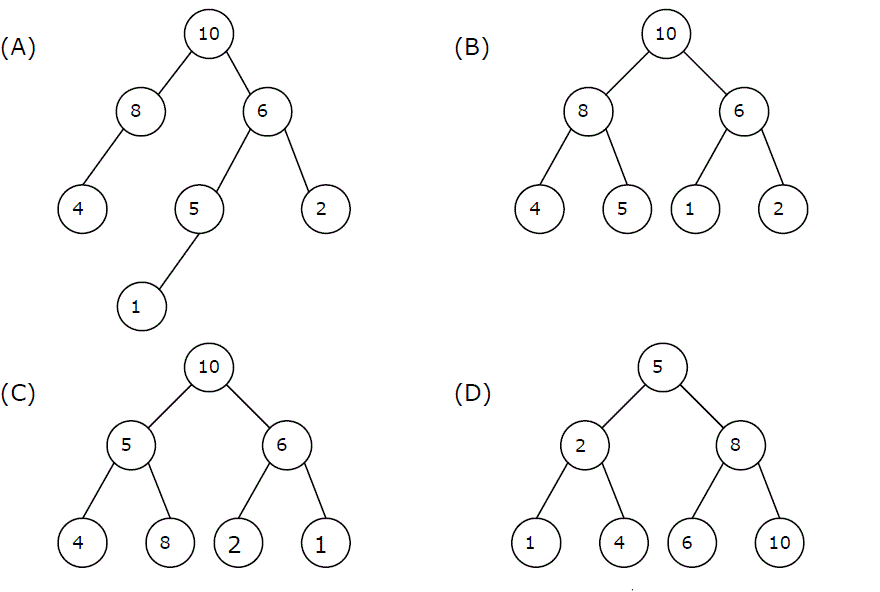
If we have two algorithms Alg1 and Alg2, and Alg1 takes time O(N) while Alg2 is O(N3), then Alg1 **always runs faster**than Alg2, for any input.

Question 5 options:

|  |  |
| --- | --- |
|  | True |
|  | False |

**Question 6 (Mandatory) (1 point)**

Which binary tree represents a max-heap?



Question 6 options:

|  |  |
| --- | --- |
|  | B |
|  | D |
|  | C |
|  | A |

**Question 7 (Mandatory) (1 point)**

Think of an algorithm that uses a Stack to efficiently check for unbalanced brackets. What

is the maximum number of characters that will appear on the stack at any time when the

algorithm analyzes the string **([]()[()])** ?

Question 7 options:

|  |  |
| --- | --- |
|  | 5 |
|  | 6 |
|  | 3 |
|  | 4 |
|  | None of these is correct |

**Question 8 (Mandatory) (1 point)**

Question 8 options:

Searching for an element in a well balanced binary search tree with n nodes takes O(



)

Searching for an element in a unbalanced binary search tree with n nodes takes O(



)

**Question 9 (Mandatory) (1 point)**

Consider the following sequence of stack operations:

**push(d), push(h), pop(), push(f), push(s), pop(), pop(), push(m)**

Assume the stack is initially empty, what is the sequence of popped values, and what is

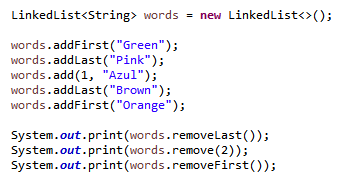
the final state of the stack?  (Identify which end is the top of the stack.)

Question 9 options:

|  |  |
| --- | --- |
|  | Sequence of popped values: h,s,f.  State of stack: m, d |
|  | Sequence of popped values: d,h,f.  State of stack: s, m |
|  | Sequence of popped values: m,s,f. State of stack: h, d |
|  | None of the options is correct |

**Question 10 (Mandatory) (1 point)**

What will the following code print out?



Question 10 options:

|  |  |
| --- | --- |
|  | BrownBrownOrange |
|  | BrownPinkOrange |
|  | BrownAzulGreen |
|  | BrownAzulOrange |

**Question 11 (Mandatory) (1 point)**

Does an instance of HashMap has two parameters that affect its performance: ***initial capacity*** and ***load factor****?*

The ***capacity*** is the number of buckets in the hash table, and the initial capacity is simply the capacity at the time the hash table is created. The ***load factor*** is a measure of how full the hash table is allowed to get before its capacity is automatically increased.

Question 11 options:

|  |  |
| --- | --- |
|  | True |
|  | False |

**Question 12 (Mandatory) (1 point)**

Given an array-based heap consisting of:

8 4 7 3 1 5 2

What is the heap that remains after a value of 6 is added to the heap?

Question 12 options:

|  |
| --- |
|  |
| 3 ->8 |
|  |

|  |
| --- |
| 4 ->4 |
|  |

|  |
| --- |
| 2-> 7 |
|  |

|  |
| --- |
| 8-> 1 |
|  |

|  |
| --- |
| 6 ->2 |
|  |

|  |
| --- |
| 5->6 |
|  |

|  |
| --- |
| 1->5 |
|  |

|  |
| --- |
| 7->3 |

**Question 13 (Mandatory) (1 point)**

Consider the following sequence of queue operations:

**enqueue(d), enqueue (h), dequeue(), enqueue (f), enqueue (s), dequeue (), dequeue (), enqueue (m)**

What would be the sequence of dequeued values, and what would be the final state of the queue? (Identify which end is the front of the queue.)

Question 13 options:

|  |  |
| --- | --- |
|  | Sequence of dequeued values: d,h,f. State of queue: s,m |
|  | None of the options is correct |
|  | Sequence of dequeued values: m,s,f.  State of stack: d, h |
|  | Sequence of dequeued values: h,s,f. State of stack: m,d |