CSC 225 SPRING 2019
ALGORITHMS AND DATA STRUCTURES I
FINAL EXAMINATION
UNIVERSITY OF VICTORIA

1. Student ID: ____________________
2. Name: ______________________
3. **Date:** 17 April 2019
   Duration: 120 minutes
   Instructor: Ali Mashreghi
4. This question paper has 16 pages, double-sided, including the cover page.
5. This question paper has 44 questions: 40 multiple choice questions and 4 written questions.
6. Questions 1 to 40 should be answered in the bubble sheet.
7. Question 41 to 44 should be answered in this exam sheet.
8. Read through all the questions and answer the **easy questions first**.

Question 1 to 20 (5 marks each)	
Question 21 to 40 (2 marks each)	
Question 41 (15 marks)	
Question 42 (15 marks)	
Question 43 (20 marks)	
Question 44 (10 marks)	
TOTAL (200) =	

PART 1: Questions 1 to 40 are multiple choice questions and should be answered in the bubble sheet.

Asymptotic complexity: For each of the pseudocodes written for the procedure `Func` with input n, what is the asymptotic time complexity? There is only one correct choice for each question.

Note: / means division, and * means multiplication

Question 1:

```
Func(n)
i = 1
while i < n
    i = i + 1
```

A. $\Theta(1)$
B. $\Theta(\log n)$
C. $\Theta(n)$
D. $\Theta(n^2)$

Question 2:

```
Func(n)
x = 1
for i = 1 to n
    j = 1
    while j < n
        j = j * 2
```

A. $\Theta(\log n)$
B. $\Theta(n)$
C. $\Theta(n \log n)$
D. $\Theta(n^2)$

In questions 3 to 5, solve the recurrences using the Master theorem. If Master theorem does not apply to a certain recurrence, choose **"master theorem does not apply"**.

Question 3:

$T(n) = 4T\left(\frac{n}{4}\right) + n^3$

A. $\Theta(n)$
B. $\Theta(n \log n)$
C. $\Theta(n^2)$
D. $\Theta(n^3)$

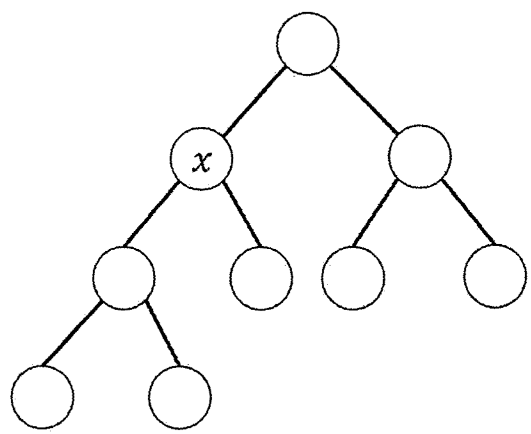
Question 4:
$T(n) = 17\,T\left(\frac{n}{4}\right) + n^2$

A. $\Theta(\log n)$
B. $\Theta(n^2)$
C. $\Theta(n^{\log_4 17})$
D. Master theorem does not apply

Question 5:
$T(n) = 4T\left(\frac{n}{4}\right) + n$

A. $\Theta(\log n)$
B. $\Theta(n)$
C. $\Theta(n \log n)$
D. $\Theta(n^2)$

Answer questions 6 and 7 for the given binary tree:



Question 6: What is the height, and the level of the node x? (The first number is height, and the second number is level)

A. 1, 2
B. 2, 1
C. 3, 2
D. 2, 3

Question 7: How many internal nodes does this tree have?
A. 3
B. 4
C. 5
D. 9

Question 8: We want to sort a list of ordered pairs (x, y). We know that there are n such pairs and we also know that x and y each are an integer between 0 and $n - 1$. For example, for $n = 4$, an example of one such list is as follows:

(1, 2)
(3, 0)
(0, 4)
(2, 3)

What is the fastest algorithm to sort such list of pairs that also uses only $O(n)$ memory? (We are allowed to see the values of the keys.)

A. Quicksort
B. Mergesort
C. Counting-sort
D. Radix-sort

Question 9: The following is the pseudocode for basic counting sort. What is the running time of this algorithm?

A. $\Theta(k^2)$
B. $\Theta(n^2)$
C. $\Theta(kn)$
D. $\Theta(n + k)$

```
BASIC-COUNTING-SORT(A, k)
Allocate array C[1..k]
for i = 1 to A.length
    C[A[i]] = C[A[i]] + 1
h = 1
for i = 1 to k
    for j = 1 to C[i]
        A[h] = i
        h = h+1
```

Question 10: In linear probing, we use a hash function as below. Assuming that m = 7 and for a given key k, $h'(k) = 3$, what is the probe sequence generated for this specific key?

$h(k, i) = (h'(k) + i) \textbf{ mod } m$

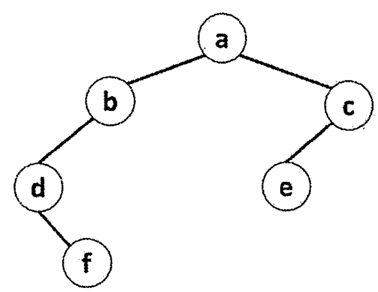
A. <3, 4, 5, 6, 3, 1, 3>
B. <0, 1, 2, 3, 4, 5, 6>
C. <0, 3, 6, 2, 5, 1, 4>
D. <3, 4, 5, 6, 0, 1, 2>

Question 11: Which pair of keys will result in a collision if we use the following hash function?

$$h(k) = (2k + 3)\ mod\ 5$$

A. (3, 4)
B. (4, 7)
C. (4, 9)
D. (5, 4)

Question 12: What is the **pre-order walk** for the following binary tree?

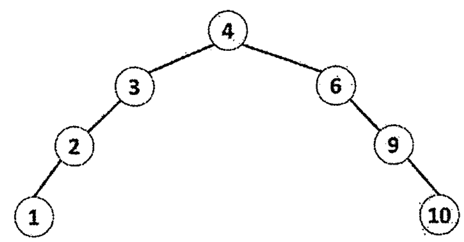


A. a, d, b, f, e, c
B. f, d, b, e, c, a
C. d, f, b, a, e, c
D. a, b, d, f, c, e

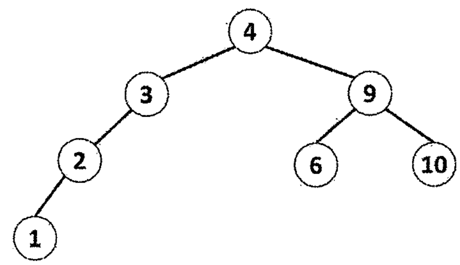
Question 13: Assume that we insert the following keys (in order from left to right) into a binary search tree that is initially empty. Which of the following will be the final binary search tree?

keys: 4, 3, 2, 1, 9, 10, 6

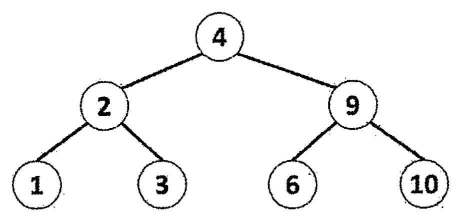
A.



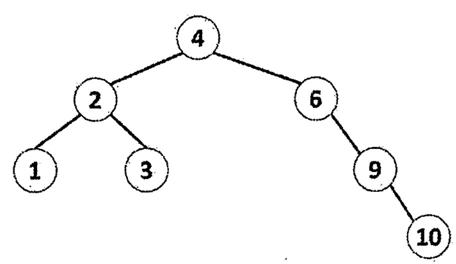
B.



C.

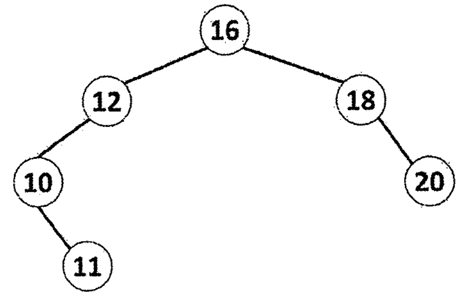


D.



Question 14: What rotation(s) should we use to fix the following AVL tree so that the balance factor is either -1, 0, or 1? (Right now, balance factor of node 12 is -2.)

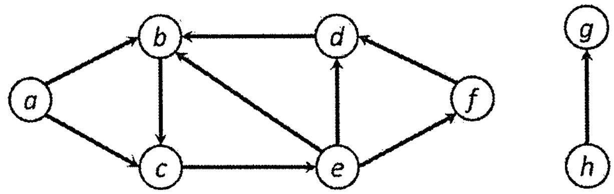
A. **right-rotate**(12)
B. **right-rotate**(10)
C. **left-rotate**(10), and then **right-rotate**(12)
D. **left-rotate**(11), and then **right-rotate**(10)



Answer the questions 15 and 16 for the following directed graph:

Question 15: What is the in-degree of node *d*?

A. 1
B. 2
C. 3
D. 4

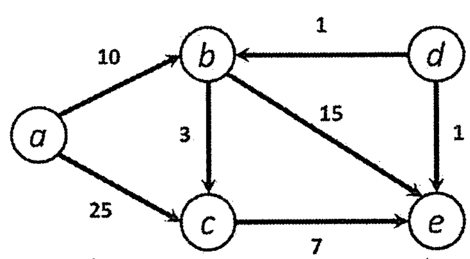


Question 16: What is the length of the shortest path from node *a* to node *f*?

A. 2
B. 3
C. 4
D. 5

Question 17: In the given graph, what is the length of the minimum weight shortest path from node *a* to node *e* with <u>at most 2</u> edges?

A. 12
B. 20
C. 25
D. 32



Question 18: In a directed graph with n nodes, what is the <u>maximum</u> number of edges on any path between two nodes?

A. 0
B. 1
C. $n-1$
D. n

Question 19: What is the minimum and the maximum number of edges in a **simple graph** (undirected and without self-loops)?

A. $n-1, n-1$
B. $0, n-1$
C. $n-1, \binom{n}{2}$
D. $0, \binom{n}{2}$

Question 20: What is the minimum and maximum number of edges in a **forest** graph?
A. $n-1, n-1$
B. $0, n$
C. $0, n-1$
D. $0, \binom{n}{2}$

Questions 21-40: [2 marks each] For each of the following data structures say how much time is required to perform the operations listed in the table.

- Imagine that there are currently n elements in the data structure with no duplicates.
- Use the most accurate big-O notation to answer. For example, if some operation takes at most $\log n$ time, you should choose $O(\log n)$ not $O(n)$.

✓ For each question do as follows:
If your answer is $O(1)$, pick choice A.
If your answer is $O(\log n)$, pick choice B.
If your answer is $O(n)$, pick choice C.

Search(k): Returns the element whose key is equal to k.
Insert(x): Inserts element x into the data structure.
Delete(x): Deletes element x from the data structure <u>assuming that a pointer to (or the index of) the element is available</u>.

Note 1: In case of the hash table consider the expected time complexity.
Note 2: In case of the sorted doubly linked list, we also keep a pointer to the last element.

	Search(k)	Insert(x)	Delete(x)	Maximum()
Max-Heap	(Q21)	(Q22)	(Q23)	(Q24)
AVL Tree	(Q25)	(Q26)	(Q27)	(Q28)
Sorted Array	(Q29)	(Q30)	(Q31)	(Q32)
Hash Table with Chaining	(Q33)	(Q34)	(Q35)	(Q36)
Sorted Doubly Linked List	(Q37)	(Q38)	(Q39)	(Q40)

PART 2: For questions 41 to 44 write your answers in the exam sheet. In case that you need extra space ask for an answer sheet from the invigilator.

Question 41: [15 marks] The pseudocode on the right is for the Paranoid-Quicksort algorithm. Paranoid-Quicksort is a randomized algorithm and has an expected worst-case running time of $O(n \log n)$. However, it is only randomized because it is using Paranoid-Partition.

```
PARANOID-QUICKSORT(A, p, r)
  if p < r
      q = PARANOID-PARTITION(A, p, r)
      PARANOID-QUICKSORT (A, p, q − 1)
      PARANOID-QUICKSORT (A, q + 1, r)
```

We know that we have the following two algorithms:

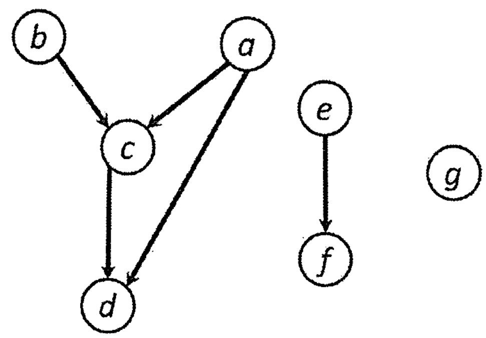
- **Select(*A, p, r, i*):** A **deterministic** algorithm that given a subarray A[p..r] returns the i^{th} smallest element in the subarray. It takes $O(r - p)$ time.
- **Partition(*A, p, r*):** A **deterministic** algorithm that partitions the subarray A[p..r] using A[r]. This algorithm also takes $O(r - p)$ time.

Using these algorithms write a pseudocode for a **deterministic version of Quicksort** that has a time complexity of $\boldsymbol{O(n \log n)}$.

Question 42:

A) [10 marks] On the graph below, compute and write down the start and finish times of visiting the vertices using a DFS. (When there are multiple options you can pick vertices in any arbitrary order that you want.)

B) [5 marks] Use part (A) to find a valid topological ordering of the nodes.



Question 38: [20 marks]
Assume that we have a simple **directed** graph G. (G could have self-loops but it has no multi-edges.) We run the DFS algorithm on G and the result is a directed tree. As we defined in class, a **back edge** is an edge from a node to its ancestor in this tree.

A) [10 marks] Prove that if there is a cycle in G then at least one edge will be classified as a back edge with respect to the tree of the DFS algorithm.

B) [10 marks] Write a complete pseudocode (including the DFS algorithm) that returns **TRUE** if G has a cycle and **FALSE** otherwise. You can assume that *Adj* is an array of lists that keeps the graph as an adjacency list.

✓ You will receive no mark if you only write a DFS algorithm without trying to solve the problem.

Question 39: [10 marks] We have the following recursive formula for $F(n)$:

$$F(n) = \begin{cases} 1 & n \leq 3 \\ F(n-1) + 2 \cdot F(n-2) + 3 \cdot F(n-3) & n > 3 \end{cases}$$

Write a **dynamic programming algorithm** that computes $F(n)$. You can use either the top-down or the bottom-up approach.

- ✓ No mark is given to an algorithm that does not follow the dynamic programming paradigm.

END OF EXAM