Data Structures and Algorithms 31251
Assessment Task 3 – Take Home Examination

Autumn 2022

Instructions

This examination is made available online at **9:30** on **June 9 2022**.

Your completed answer file is due at **9:30** on **June 10 2022** and must be submitted online via https://canvas.uts.edu.au/courses/22030/assignments/90795 on Canvas.

There are **15 multiple-choice questions and 6 short-answer questions**. Your answer to each question attempted should commence after the question number in the answer file.

The examination is worth **30%** of the marks available in this subject. The contribution each question makes to the total examination mark is indicated in marks.

This examination is an open book examination.

This examination is expected to take approximately **2 hours** of working time. You are advised to allocate your time accordingly. Your answer file may be submitted at any time before the due time. Please allow time to complete the submission process.

Please submit your file in PDF/Word format unless directed otherwise. Please name your file as follows:

EXAM_31251_student number   e.g. EXAM_31251_12345678

Word Limit

There is a word limit for each question. The most important thing is to answer the question in a succinct manner. This means that your answer can consist of a word count less than the imposed word limit. A ten percent (10%) leeway on word counts is permitted.

Your answers should not contain any footnote/reference.

Important Notice – Exam Conditions and Academic Integrity

In attempting this examination and submitting an answer file, candidates are undertaking that the work they submit is a result of their own unaided efforts and that they have not discussed the questions or possible answers with other persons during the examination period. Candidates who are found to have participated in any form of cooperation or collusion or any activity which could amount to academic misconduct in the answering of this examination will have their marks withdrawn and disciplinary action will be initiated on a complaint from the Examiner.

Exam answers must be submitted **via Turnitin**. Staff may ask that a student undertake an oral test to ensure they have completed the work on their own and to assess their knowledge of the answers they have submitted.

Students must not post any requests for clarification on the Discussion Boards on Blackboard, Canvas or Microsoft Teams. Any requests for clarification should be directed by email to **Xianzhi Wang** on **Xianzhi.Wang@uts.edu.au**. Where clarification is required it will be broadcast by email to all students in the exam group. **Xianzhi Wang** will be available via email during the first hour and the last hour of the examination time window.

Question 1 (0.8 Mark)

What is the time complexity of inserting a new element before the fourth to last element in a singly linked list? Suppose the length of the linked list is n.

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

Question 2 (0.8 Mark)

Suppose you use a simple division hash function, `h(k)=k%n`, and linear probing to insert a series of elements: `4,5,6,8,10`, one by one to an array (Suppose the length of array: n=6). What would be the array position (or index) to store `10`?

A. 0
B. 1
C. 3
D. 4

Question 3 (0.8 Mark)

What is the worst-case time complexity of adding an element to a binary heap? Suppose n is the number of elements that already exist in the heap before the addition. You may consider either a max-heap or a min-heap; that will not affect the result.

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

Question 4 (0.8 Mark)

The visiting orders of a Postorder traversal and an Inorder traversal of a tree are shown below:

Post-order: `B D F A C E G`

In-order: `G E D B C F A`

What is the visiting order a Preorder traversal of the same tree?

A. `G E A F C D B`
B. `G E C D B A F`
C. `G E D B C F A`
D. None of the above

Question 5 (0.8 Mark)

Suppose `x` is the value at the bottom of a `std::stack` after the following operations: `push(1)`, `push(3)`, `push(5)`, `pop()`,`pop()`, `push(7)`, `pop()`, `push(9)`. What operation sequences make the value at the front of a `std::queue` equal `x` (More than one answer may be selected)?

A. `push(1)`, `push(3)`, `pop()`, `push(1)`, `push(5)`, `pop()`, `push(1)`, `push(7)`
B. `push(1)`, `push(3)`, `push(1)`, `push(5)`, `pop()`, `pop()`, `pop()`, `push(7)`, `push(9)`
C. `push(5)`, `push(3)`, `push(1)`, `pop()`, `pop()`, `push(9)`, `push(3)`, `push(7)`, `pop()`
D. `push(2)`, `pop()`, `push(2)`, `push(7)`, `push(1)`, `pop()`, `push(9)`, `push(4)`, `pop()`

Question 6 (0.8 Mark)

Which of the data structures below can be iterated over using `std::iterator`? (More than one answer may be selected)? To iterate over a data structure means to access all the elements in the data structure one by one.

A. `std::tuple`
B. `std::map`
C. `std::set`
D. `std::list`

Question 7 (0.8 Mark)

Which of the following is TRUE about `pointer`s and memory deallocation in C++?

A. We cannot use the '`delete`' keyword to deallocate the space that is allocated statically.
B. Suppose `head` is a pointer to the first node of a linked list. We can do `delete head;` to deallocate the memory taken by the whole linked list.
C. Suppose `p` is a pointer to certain memory space. We can simply set `p=nullptr;` to deallocate the space.
D. We do not need to deallocate space as C++ can automatically recycle the space that is not in use.

Question 8 (0.8 Mark)

Which of the following is/are TRUE about hash functions (More than one answer may be selected)?

A. A "good" hash function can generate a hash according to the input very quickly.
B. A "good" hash function may have a certain chance of generating different hashes for the same input.
C. A "good" hash function has a low chance of generating the same hash for different inputs.
D. A hash function may work well for one set of inputs but not for another set of inputs.

Question 9 (0.8 Mark)

Suppose a binary heap (Heap) and a binary search tree (BST) contain the same elements (e.g., `int`s). Which of the following is/are NOT true about binary heap and binary search tree?

A. The height of the Heap might be larger than the height of the BST.
B. Heap-sort may have lower big-O complexity in giving an (either ascending or descending) ordering of the elements than the BST does.
C. It is more efficient to do a lookup in BST than in Heap.
D. Heap always takes less memory space than BST no matter what data structures are used to implement them.

Question 10 (0.8 Mark)

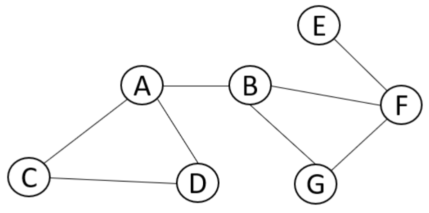
What is the time complexity of the following code (Suppose `m>0` and `n>1`)?

```
int func (int m, int n){
    int i=0;
    while (m<100){
        i++;
        m++;
    }
    for (int i=0; i<n-1; i++){
        for (int j=n; j>i; j--)
            return i*j;
    }
}
```

A. $O(m)$
B. $O(n^2)$
C. $O(m+n^2)$
D. $O(1)$

Question 11 (0.8 Mark)

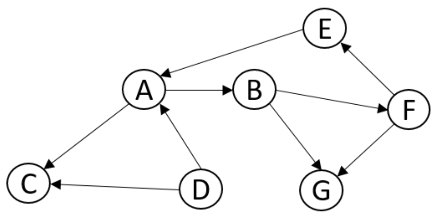
How many articulation points (i.e., cutting vertices) are there in the graph below?



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

Question 12 (0.8 Mark)

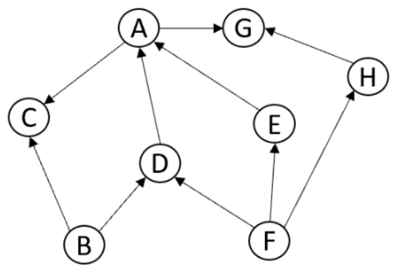
How many strongly connected components (SCCs) are there in the graph below?



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

Question 13 (0.8 Mark)

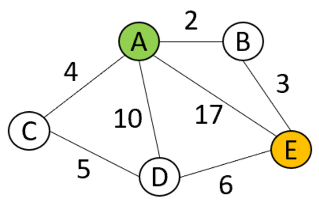
Which of the following is a possible result of topological sorting (regardless of which topological sorting algorithm you use) on the graph below (More than one answer may be selected)?



A. B, C, F, D, H, E, A, G
B. B, F, D, E, A, C, H, G
C. F, H, B, D, E, A, C, G
D. B, F, D, E, A, G, C, H

Question 14 (0.8 Mark)

Suppose you are using Dijkstra's algorithm to find the shortest path from `A` to `E`. Which of the following statement is/are TRUE about the distances of nodes as calculated by the algorithm (More than one answer may be selected)? ‘Distance’ of a node is the weight of the current path from `A` to the node.



A. The distance of `A` is always `0` and will not be updated by the algorithm.

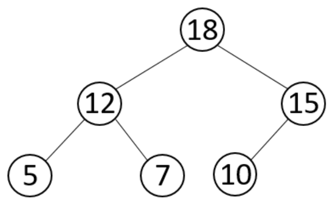
B. The distance of `E` should not no longer be positive infinity once the algorithm updates the distance of `E`.

C. The distance of `D` is initialised to positive infinity by the algorithm.

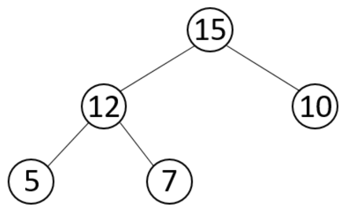
D. The algorithm terminates immediately once the distance of E is updated.

Question 15 (0.8 Mark)

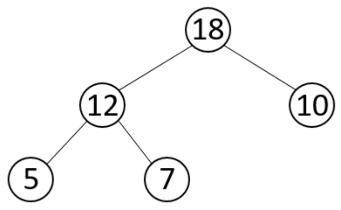
Given the max-heap below, what will the tree structure look like after we remove one element (i.e., the maximum element) from it?



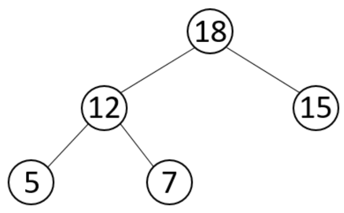
A.



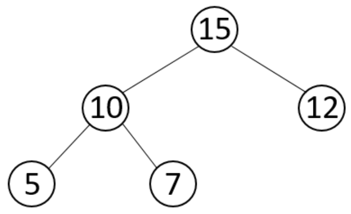
C.



B.



D.



Question 16 (3 Marks)

Considering a collection of elements `{3,7,6,2,9,4}`, what insertion order of these elements would produce a Binary Search Tree (BST) that has the smallest height?

Please give at least four insertion orders that meet the above requirement (word limit: 100).

Question 17 (3 Marks)

Can you briefly describe what the function below does in plain English (word limit: 100)?

```
class Node{
    public:
        int data;
        Node* next;
};

int func(Node* node, int number){
    if(node){
        if(node->data > 0)
            return func(node->next, number) - node->data;
        else if(node->data < 0)
            return func(node->next, number) + node->data;
        else
            return func(node->next, number + node->data) + number;
    }
    return 0;
}
```

Question 18 (3 Marks)

Suppose we have a square matrix: `int matrix[n][n]`, where `n>0`. Can you briefly describe what the code below does (word limit: 100)? You may use an example and draw a figure to illustrate your idea if you want.

```
for (int x = n-1; x > -1; x--) {
    for (int y = 0; y <n; y++) {
        if (x < n-1 && y > 0) {
            matrix[x][y] += std::min(matrix[x+1][y], matrix[x][y-1]);
        } else if (x < n-1) {
            matrix[x][y] += matrix[x+1][y];
        } else if (y > 0) {
            matrix[x][y] += matrix[x][y-1];
        }
    }
}
std::cout << matrix[0][n-1] << std::endl;
```

Question 19 (3 Marks)

Suppose we wish to implement a set that can facilitate the fast lookup of `int`s. Suppose the number of `int`s to be stored is no more than n^2, and the `int`s do not equal each other. We have decided to use a two-dimensional array `int A[n][n]` as the underlying data structure to store the `int`s.

Our current mechanism is based on a simple division function `h(k)=k%n` and linear probing. Suppose we are to insert an `int` value (say x) to the set. We first calculate `x%n` and then attempt to store x to the array cell `A[x%n][x%n]`. If `A[x%n][x%n]` is occupied, we check

`A[(x%n+1)%n][(x%n+1)%n]`, `A[(x%n+2)%n][(x%n+2)%n]`, …, `A[(x%n+n-1)%n][(x%n+n-1)%n]`

one by one until an empty spot is found to store x or all those cells have been checked.

Can you identify the problem with the above insertion approach for inserting `int`s to the set? Can you design an insertion mechanism that potentially allows for a faster looking up of `int`s from the two-dimensional array in practice?

Please briefly describe your method in plain English, pseudo-code, or C++ code (word limit: 200).

Question 20 (3 Marks)

Suppose we have a sequence of numbers: `1,8,5,2,6,3,9,7,4,2,3`. We aim to find a longest turbulence in the sequence. 'Turbulence' is a consecutive sub-sequence where the numbers rise and drop alternately. Every sub-sequence of a turbulence is also a turbulence. For example,

- `1` and `8` and `5` are all turbulences because each of them contains only one number.
- `1,8` and `8,5` are both turbulences because `1` rises to `8` in the first sequence and `8` drops to 5 in the second.
- `1,8,5` is a turbulence because `1` rises to `8` and then drops to `5`.
- `8,5,2` is not a turbulence because `8` drops twice (first to `5` and then to `2`).
- The longest turbulence in the given sequence is `5,2,6,3,9,7`.

Can you design a brute-force algorithm to find the longest turbulence in a given sequence of numbers (such as the sequence provided above)? What is its time complexity in terms of big-O? (word limit: 200)

Question 21 (3 Marks)

Can you describe an algorithm that is likely to achieve better running time than brute-force for Question 20? (word limit: 200).