THE UNIVERSITY of ADELAIDE

Student ID: a1906808
Family Name: Tang
Other Name: Piseth
Desk No: N49

Primary Examination, Semester 1, 2022

107592 Algorithm Design and Data Structures COMP SCI 2103, 7103

Course Coordinator: Dr Cheryl Pope

Total Duration: 120 mins

Questions	Time	Marks
Answer all 8 questions	120 mins	120 marks
		120 Total

Instructions for Candidates

- This is a Modified Closed-Book examination (refer to Materials)
- Answer all questions in the space provided on the exam paper
- If you need more space, use the extra blank pages at the end of the exam
- Write your desk number on each page
- Examination material must not be removed from the examination room

Materials permitted:

- Paper dictionary - general or translation
- Two A4 double sided sheets of notes

DO NOT COMMENCE WRITING UNTIL INSTRUCTED TO DO SO

STOP WRITING IMMEDIATELY WHEN INSTRUCTED

Name:_____

Inheritance and Object Oriented Programming

Question 1

(a) Explain how the function *getMin()* in the class below demonstrates *encapsulation/information hiding*

```
class myClass {
    // list of numbers stored as strings
    std::vector<std::string> myNumbers;

    public:
        int getMin(); // return the smallest number in the list
};
```

Recall that encapsulation usually refers to the bundling of data and method within a single unit or class and that is in-line with information-hiding as well. Indeed, the class myClass encapsulates the data 'myNumbers' (vector) and method 'getMin()'. That way, the user or 'client' doesn't need to worry how the internal data is stored. From their perspective, they only know that invoking the getMin() method returns the smallest number in the data, other details (data storage, getMin()'s logic) are hidden.

[3 marks]

(b) Explain the difference between virtual and a pure virtual function (i.e. *virtual myFunction()* vs *virtual myFunction() = 0*). In your answer include an explanation of when each should be used and the effect on child classes.

The differences are illustrated as follows:

1). Virtual Functions:

- Use case: when one wants to provide a common interface with a default behavior that can be overridden by derived classes.
- Effect on child classes: derived classes have the options of overriding the virtual function, else the class's version shall be used.

[3 marks]

2). Pure virtual functions:

- Use case: when one wants to enforce a contract that all derived classes must provide their own implementation...
- Effect on child classes: derived classes are required to override the pure virtual function. Else, they become abstract and instantiating them causes compile-time errors.

Name:_____

(c) The code below prints nothing when we want it to print "Roar". Explain what is wrong and how to correct the code.

```
class Animal {
    public:
        virtual void makeSound() {
             std::cout << "";
        }
};

class Tiger : public Animal {
    public:
        void makeSound() {
            std::cout << "Roar";
        }
};

int main (void) {
    Animal zoo[10];
    int zooSize=0;

    Tiger tigger;
    zoo[zooSize] = tigger;
    zooSize++;

    for (int i=0; i<zooSize; i++) {
        zoo[i].makeSound();
    }
}
```

• The code attempts to implement dynamic-binding or run-time polymorphism where the types of objects are known only at run-time. The bug is in not using pointers or references (to derived classes) to implement run-time binding. Put it simply, in C++, pointers or references are required to implement run-time polymorphism due to the way the language handles "Object Slicing" and "dynamic dispatch". We need only fix the main code as follows:

```
Animal * zoo;
...
zoo[size] = &tiger;
...
```

[3 marks]

[Total for Question 1: 9 marks]

Name:_____

Recursion

Question 2

(a) Even if a recursive function has a base condition, it may not successfully terminate. Explain how this could happen?

- The base case was flawed to begin with. Incorrect base cases occur very frequently.
- We didn't subdivide our problem appropriately. Maybe we didn't at all, in that case causing an infinite loop or we did but in a way that jumps over the base case (in-line with the first point).

[2 marks]

(b) Write a *head recursive* C++ function to compute the factorial of n: $n! = n * (n-1) * (n-2) .... * 1$

```
int head_fact(int n) {
    if (n < 1) : return 1;
    return n * head_fact(n-1);
}
```

[4 marks]

Name:_____

(c) Write a *tail recursive* C++ function to compute the cumulative sum of n: $sum = n + (n-1) + (n-2).... + 0$

```
int sum(int n, int acc=0){
    if (n < 1) return acc;
    else return sum(n-1, acc + n)
}
```

[5 marks]

[Total for Question 2: 11 marks]

Name:_____

Linked Lists, Stacks, Queues

Question 3

Let class Node be defined as follows, and consider a linked list class that uses Node objects for storing the items. The linked list class has a member variable ***Node * head***, that points to the first node of the list.

```
class Node {
    private:
       int data;
       Node *link;
};
```

(a) Linked list must have access to the data and link variables of the Nodes in the list, explain what change to the Node class declaration should be made to allow a LinkedList class to access these without making them public to other classes.

[2 marks]

- We can make two modifications to the Node class declaration to allow a LinkedList class to access the Node's data.

1). Implement getters and setters, one for each attribute, data and link.

2). Or, we could use the friend keyword which does the job. We write `friend class LinkedList;` in Node class as follows:

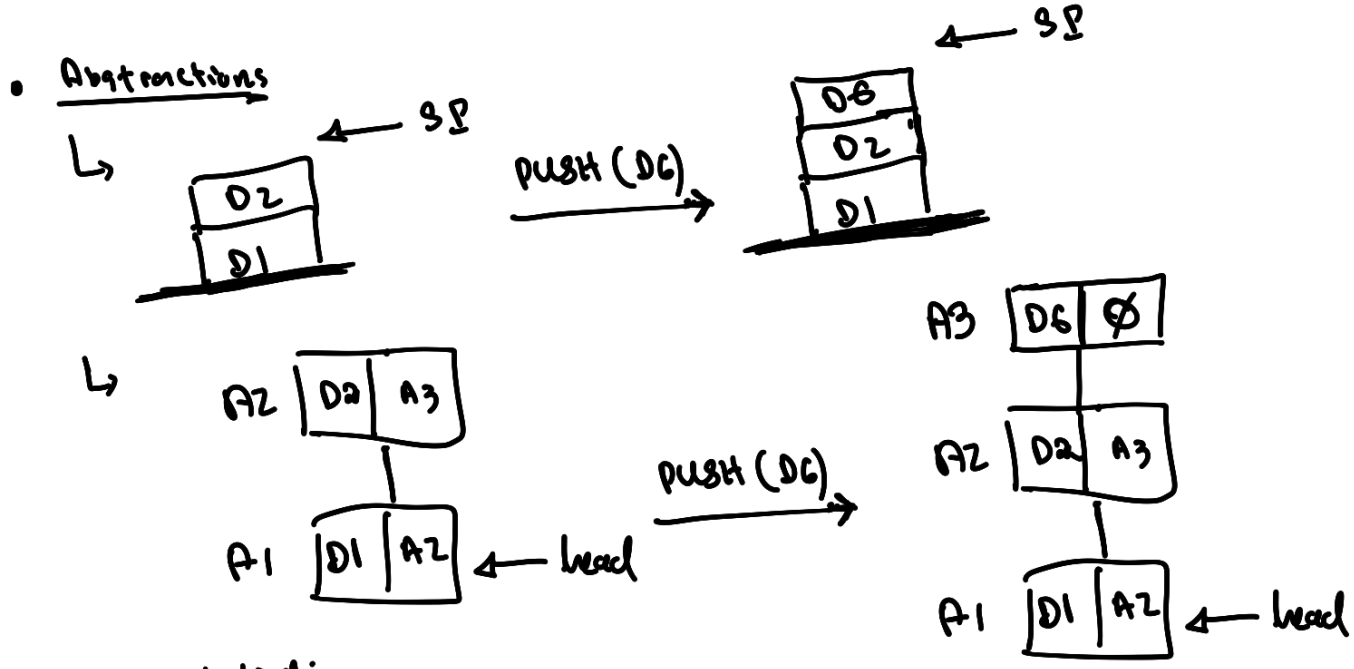
```
class Node {
private:
    int data;
    Node* link;
    friend class LinkedList;
};
```

```
class LinkedList {
private:
    Node* head;
// LinkedList has access
 to Node's private data
};
```

NOTE: One can either declare the friend keyword in a private section or a public section.

Name:_____

(b) Write the C++ push function for a Stack implemented as a linked list.



- Implementation:

```
void PUSH(int data) { // This is just like push_back(int data)
    Node* nodeToBeInserted = new Node(data, nullptr);
    Node* curr = head;
    if (curr == nullptr) {
        head = nodeToBeInserted;
        return;
    }
    else {
        Node* lastNode = getLastNode();
        lastNode -> setLink(nodeToBeInserted);
    }
```

$\Theta(1)$ | $\Theta(n)$

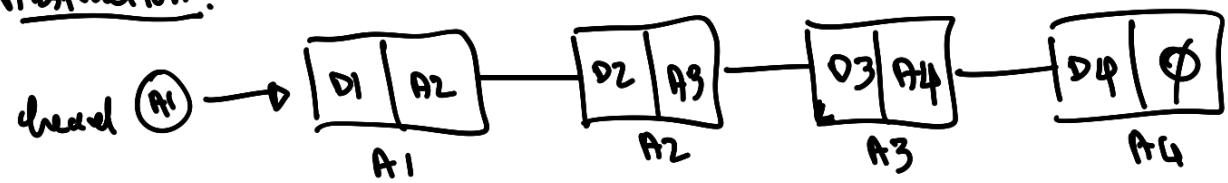
[4 marks]

- $T(n) \in \begin{cases} \Theta(1), & \text{best case} \\ \Theta(n), & \text{average \& worst case} \end{cases}$
- $S(n) \in \Theta(1)$ (all cases)

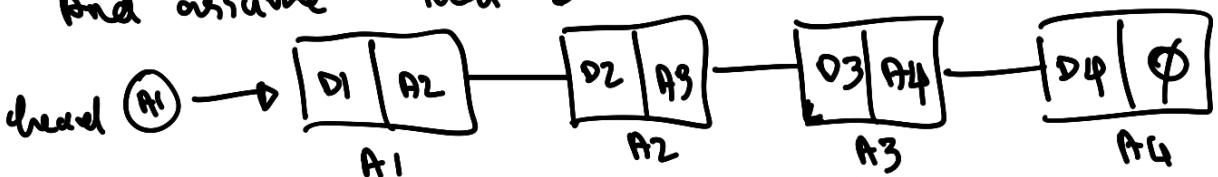
Name:_____

(c) Write C++ code for a member function of linked list class ***void RemoveMin()***, that removes the node with the smallest value stored in the linked list.

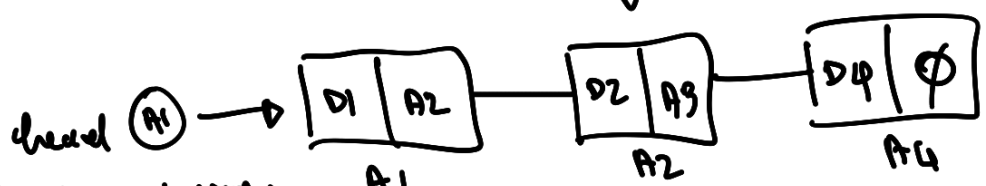
Abstraction:



Suppose there are no duplicates, i.e., $\forall i \neq j : D_i \neq D_j$.
And assume that D3 is the smallest element. Then,



RemoveMin()



Implementation:

```
void RemoveMin() {
  int temp = 0;
  Node *curr = head;
  Node* currMinNode = head;
  temp = curr->getData();
  if (curr = nullptr) return;                          | Θ(1)
  else if (curr->getLink() == nullptr) DELETE(currMinNode); | Θ(1)
  while (curr != nullptr) {
    if (curr->getData() < temp) {
      temp = curr->getData();
      currMinNode = curr;
    }
    curr = curr->getLink();
  }                                                    | Θ(n)
  DELETE(currMinNode);                                 | Θ(1)
}
```

[14 marks]

[Total for Question 3: 20 marks]

$$T(n) \in \begin{cases} \Theta(1), & n \leq 1 \\ \Theta(n), & n > 2 \end{cases}$$

Name:_____

Complexity

Question 4

(a) Indicate if the following statement **true** or **false** and give a short explanation to justify your answer.

Ω is the best case complexity of an algorithm.

This is false. It all comes down to the definition. Ω is the lower bound for a running time of an algorithm. Unless the term "best case complexity" refers to the lower bound, then the given statement is false. Bounds (lower & upper) and scenarios (best, worst, average) are different things. For each case, there's an upper bound & lower bound. The bounds give a theoretical limit on the algorithm running time, whereas the scenarios are concerned with the input into the algorithm.

[2 marks]

(b) What is the time complexity of the following code segment? Briefly explain.

```
for (i=0; i<n; i++) {
  if (i==0) {
    [redacted]
  }
  else {
    cout << i;
  }
}
```

Θ(n) Θ(n)

Θ(1) Θ(1)

Θ(n) + Θ(1) = Θ(n)

for (i = 0 → n) {

// Θ(n) work

}

⟹ T(n) ∈ Θ(n²)

[3 marks]

(c) Find a tight upper and lower bound for function $3n^2 - 5n + 7$, i.e., find

Name:_____

$g(n)$ where $3n^2 - 5n + 7 \in \Theta(g(n))$. Also, formally prove that the following statement holds $3n^2 - 5n + 7 \in \Theta(g(n))$.

. Given $f(n) = 3n^2 - 5n + 7$.

↳ Claim: n^2 is a lower bound for $f(n)$.

Given $f(n) = 3n^2 - 5n + 7$. Observe that

$f(n) = 3n^2 - 5n + 7 \leq 3n^2 + 5n^2 + 7n^2$

$= 15n^2 \quad \forall n \geq 1$.

∴ There exist $N_0 = 1 \in \mathbb{N}$, $C = 15 \in \mathbb{R}_{>0} = \{x \in \mathbb{R} \mid x > 0\}$ such that

$f(n) \leq C \cdot n^2 \quad \forall n \geq N_0 \Rightarrow f(n) \in O(n^2)$ (1)

↳ Claim: n^2 is also an upper bound for $f(n)$.

Given $f(n) = 3n^2 - 5n + 7$.

Observe that we just need to observe the largest term, $3n^2$; and pick any coefficient smaller than its coefficient.

Clearly, $f(n) = 3n^2 - 5n + 7 \geq 2n^2$, $\forall n \geq 1$.

Then, there exist $N_0 = 1 \in \mathbb{N}$, $C = 2 \in \mathbb{R}_{>0}$ such that

$f(n) \geq cn^2$, $\forall n \geq N_0 \Rightarrow f(n) \in \Omega(n^2)$ (2)

Thus, $g(n) = n^2$.

. Proof: $f(n) \in \Theta(g(n))$ follows immediately from the definition of tight bound. $f(n) \in \Theta(g(n))$ if, and only if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$, which follows from (1) and (2) immediately. ∎

[5 marks]

[Total for Question 4: 10 marks]

Name:_____

Sorting and Searching

Question 5

(a) You want to sort a list of names, each consisting of first and last names. People may have the same first name or last name, but the combination of both is unique.

The list should be sorted by first names. For repeated first names, sort them by last names. (e.g. John Blake, John Smith)

Assume that the list is already sorted by last names. Answer the following questions.

i. Identify an appropriate sorting algorithm to complete the process, i.e. to sort by first name. Briefly justify your choice of sorting algorithm.

Any stable sorting algorithms will do. Assuming that space is not at a premium, we choose the Merge Sort procedure to sort by first name. Although, it takes $\Theta(n)$ space complexity, it takes $\Theta(n \log n)$ time complexity, which is better than insertion sort and bubble sort for large enough n, which we also assume to be the case that the database to-be sorted is huge.

[2 marks]

Name:_____

ii. Write the steps or high-level pseudocode of the sorting algorithm you identified in the previous subquestion.

```
MERGE(A, p, q, r):
    n_L = q - p + 1  || len(A[p:q])
    n_R = r - q      || len(A[r:q])
    let L = [0 : n_L - 1], R = [0 : n_R - 1]
    for i = 0 to n_L - 1 : L[i] = A[p + i]
    for j = 0 to n_R - 1 : R[j] = A[q + j + 1]
    i, j, k = 0, 0, p
    while(i < n_L and j < n_R):
        if (L[i] ≤ R[j]):
            A[k] = L[i]
            i++
        else:
            A[k] = R[j]
            j++
        k++
    while (i < n_L):
        A[k] = L[i]
        i++
        k++
    while (j < n_R):
        A[k] = R[j]
        j++
        k++
```

```
MERGE_SORT(A, p, r):
    if p ≥ r : return
    q = ⌊(p + r)/2⌋
    MERGE_SORT(A, p, q)
    MERGE_SORT(A, q+1, r)
    MERGE(A, p, q, r)
```

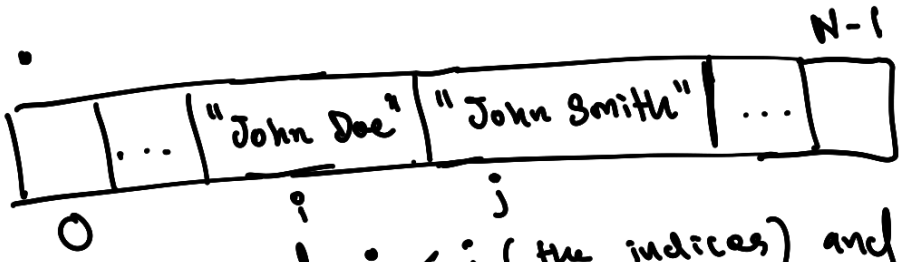
[4 marks]

iii. Identify a sorting algorithm that will **NOT** work, i.e., the list will not be

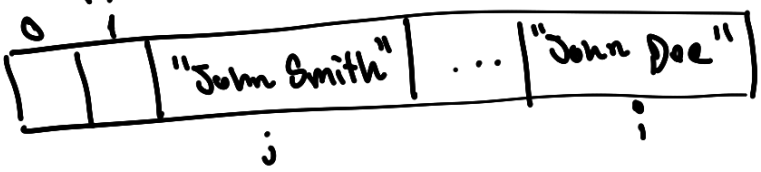
Name:_____

sorted by first then last name. Briefly justify your answer and provide an example.

- Conversely, any unstable sorting algorithms shall do. For simplicity, pick selection sort. Selection sort will not work because it does not preserve the relative order of elements with the same value.
- Example: Given A =



Observe that A is sorted by last name and $i < j$ (the indices) and $A[i] < A[j]$. We assume "<" is well-defined somewhere. Then the selection sort procedure yields



where the original $A[i] > A[j]$ instead.

[4 marks]

(b) Compare quick sort and merge sort in terms of time and space complexity (consider best/average/worst cases).

	Time complexity			Space complexity
	Best	Average	Worst	Worst
Quick sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n^2)$	$O(1)$
Merge sort	$\Omega(n \log n)$	$\Theta(n \log n)$	$O(n \log n)$	$O(n)$

[4 marks]

Name:_____

(c) Illustrate the process of sorting the list $\{1, 2, 3, 4, 5, 6, 7\}$ using quick sort to achieve the fastest runtime.

- Sort - by medium 3

[4 marks]

Name:_____

(d) You are working with a large array of data containing only single digits as values (i.e., 0,1,2,3...,8,9). The data is not ordered in any way and may contain duplicates. You must be able to determine if a given value appears in the array or not.
You decide to use binary search to improve your worst case search complexity to $O(\log n)$.

i. How must you transform this data before you can use binary search?

- One must transform the data before binary search can be used, else binary sort will return the wrong value.

[1 mark]

ii. What is the most time efficient method that we covered in this course for transforming the data? What is the complexity of this method?

- Considering that the array may contain duplicates, counting sort works well with such data distribution, whereas bucket sort doesn't and will require a large amount of memory to hold the duplicates in each bucket. Moreover, the array contains only integers, for which counting sort has spefically been designed for.

[3 marks]

- Time complexity : $T(n) \in \Theta(n+k)$; n := size of array
k := maximum element in the array.

Name:_____

iii. Illustrate the process of the binary search algorithm on the list $\{19, 15, 14, 10, 6, 5, 1\}$, start = 0, end = 6 and item = 1.

Descending Order

$$mid = \left\lfloor \frac{start + end}{2} \right\rfloor$$

start	mid	end	item = A[mid] ?
0	3	6	No
4	5	6	No
6	6	6	Yes

return mid = 6.

[3 marks]

(e) Can the time complexity of sorting algorithm be lower than $O(n)$? Brifely justify your answer.

- No. Because by inspection alone, we need to inspect all n elements to determine that the list is sorted. For comparison-based sorting procedures, the lower bound is $\Omega(n \log n)$ and even with a check such as is_sorted(A, n), that is still linear. The same argument holds for distribution-based sorts.

[2 marks]

Name:_____

(f) Identify one non-recursive, comparison-based sorting algorithm where the best and worst cases have different complexity. Explain why they differ.

- Insertion sort is a non-recursive, comparison-based sorting algorithm where the best & worst cases have different time complexity.
- To understand why, we define the best case as when the input array is already sorted and the worst case is when the array is randomly shuffled. And we look at the insertion sort implementation below

[3 marks]

[Total for Question 5: 30 marks]

```
INSERTION-SORT(A, n):
    for i = 1 → n:
        key = A[i]
        j = i - 1
        while j > 0 and A[j] > key:
            A[j+1] = A[j]
            j = j - 1
        A[j+1] = key
END
```

- Observe that when an array is sorted (best case), the second conditional statement A[j] > key yields false since A[j] (the value on the left) is already smaller than or equal to key (the value on the right to be inserted so the body of the loop is not reached leading to just $\Theta(1)$ check. Doing the check n times (each taking $\Theta(1)$) yields $\Theta(n)$, linear time complexity.

Name:_____

Trees

Question 6

(a) For each of the following statements, indicate if it is **true** or **false** and give a short explanation to justify your answers.

i. The depth and the height of a tree is the same.

No.

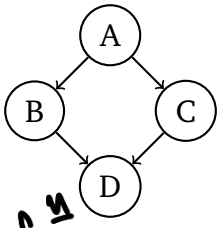
[2 marks]

ii. Any tree can be implemented with a doubly linked-list.

No ...

[2 marks]

iii. The following diagram is **NOT** a tree.

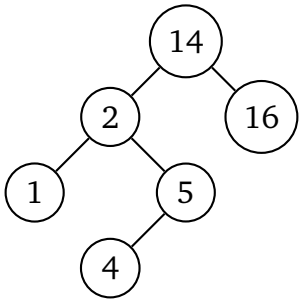


Yes. Look at def.

[2 marks]

Name:_____

(b) Consider the binary search tree shown below:



i. What is the height of the tree after inserting 3

[1 mark]

ii. Write the output when traversing the resulting tree (after inserting 3) pre-order.

[1 mark]

(c) Assume the original tree from part (a) (without inserting 3) is an AVL tree.

i. Identify all the node(s) that are unbalanced (need to be rebalanced).

[2 marks]

Name:_____

ii. Explain steps to rebalance the tree. Each step should identify:
- situation before rotation
- type of rotation necessary
- how to rotate

[4 marks]

iii. Write the output when traversing the resulting balanced AVL tree post-order.

[1 mark]

[Total for Question 6: 15 marks]

Name:_____

Heaps

Question 7

(a) An array h stores values of a binary *min* heap and currently holds the values:
1 6 5 8 9 7

i. Write a min heap array of the resulting heap after adding 3

[1 mark]

ii. Consider the starting heap (before adding 3). Write a min heap array after removing 1

[1 mark]

(b) Write a max heap array after the following numbers are inserted in order:
14,2,16,1,5,4,3

[2 marks]

Name:_____

(c) Consider the process of sorting an array $\{10, 1, 8, 9, 7, 2, 5, 3\}$ using heap sort. At each step, identify the min heap array and the sorted array.

[6 marks]

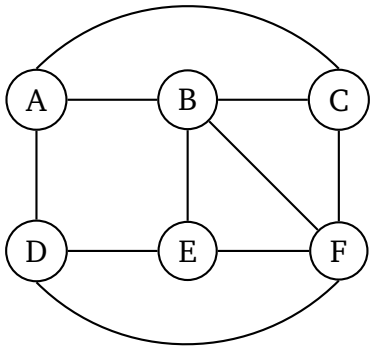
[Total for Question 7: 10 marks]

Name:_____

Graphs

Question 8

(a) Consider the following graph:



Identify an adjacency matrix that represents the above graph.

[2 marks]

Name:_____

(b) Consider an undirected graph with n nodes. Answer the following five questions.

i. What is the *space* complexity to represent such a graph using an adjacency matrix?

[1 mark]

ii. What is the *space* complexity to represent such a graph using an adjacency list?

[1 mark]

iii. What is the *time* complexity to determine whether there is an edge (v_j, v_k) for a graph represented using an adjacency matrix M? Explain the process.

[2 marks]

iv. What is the *time* complexity to determine whether there is an edge (v_j, v_k) for a graph represented using an adjacency list? Explain the process.

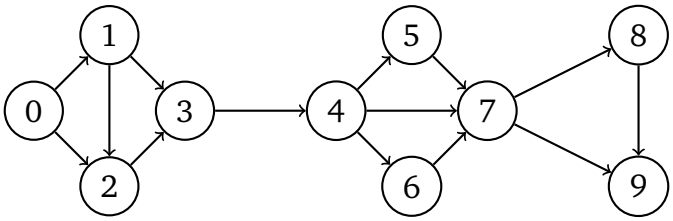
[2 marks]

Name:_____

v. Are there any advantages to representing a graph using an adjacency list over an adjacency matrix? Justify your answer.

[2 marks]

(c) Consider the following graph:



List the vertices in the order they would be visited using *Breadth First Search* starting at node 0.
Assume when choosing among nodes to visit next, the algorithm chooses the lowest number, i.e., if choosing among vertex neighbours 1, 3, and 8, the algorithm would choose 1.

[2 marks]

Name:_____

(d) Explain how to solve the eight queens problem using *Depth First Search*.

As a reminder, the eight queens problem is the problem of finding a way to place eight queens on an 8x8 chessboard so that no queens share the same row, column or diagonal.

[3 marks]

[Total for Question 8: 15 marks]

Name:_____

Extra Space for Answers
Please indicate the question number clearly

Name:_____

Extra Space for Answers
Please indicate the question number clearly

Name:_____

Extra Space for Answers
Please indicate the question number clearly

Name:_____

Extra Space for Answers
Please indicate the question number clearly

End of exam