1(a)

		Rules							
Conditions	First timer	Y	Y	Y	Y	N	N	N	N
	Spent at least $1000 in past 3 months	Y	Y	N	N	Y	Y	N	N
	Inactive for 1 month	Y	N	Y	N	Y	N	Y	N
Actions	$5 discount	X	X	X	X			X	
	10% discount	X	X			X	X		
	Additional 5% discount	X				X			
	No discount								X

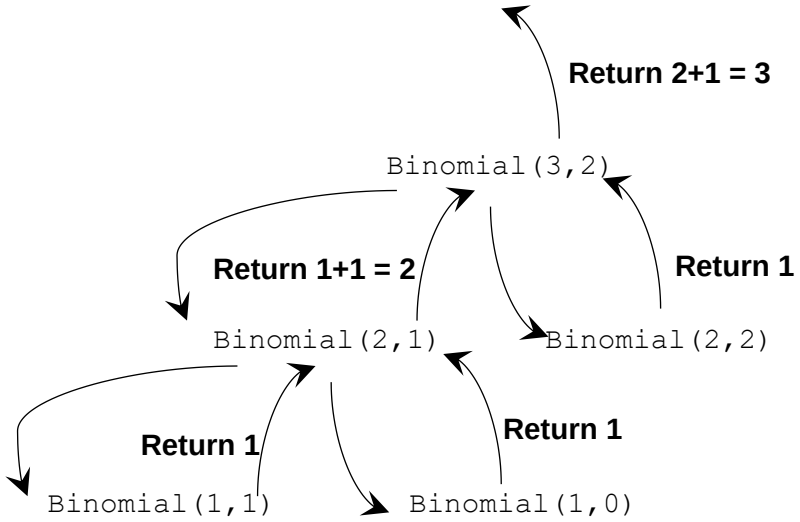
1(b)

		Rules				
Conditions	First timer	Y	N	N	N	N
	Spent $1000	N	Y	Y	N	N
	Inactive	N	Y	N	Y	N
Actions	$5 discount	X			X	
	10% discount		X	X		
	Extra 5%		X			
	No discount					X

2(a)

A recursive function is one that calls itself. Line 06 makes it recursive.

2(b)



(c) Any of the following:

- R > N

- R < 0

- N < 0

3(a)

Iteration	Answer	Queue
	0	[5, –2, 3, –1]
1	0 + 5 = 5 5 * 2 = 10	[–2, 3, –1]
2	10 – 2 = 8 8 * 2 = 16	[3, –1]
3	16 + 3 = 19 19 * 2 = 38	[–1]
4	38 – 1 = 37 37 * 2 = 74	[]

The function returns 74

3(b)

There is one extra multiplication by X at the end

3(c)

Solution 1: Swap the addition (line 04) and multiplication (line 05)

```
01  FUNCTION Evaluate(X : INTEGER, Coeffs : QUEUE) RETURNS INTEGER
02      Answer ← 0
03      REPEAT
04          Answer ← Answer * X
05          Answer ← Answer + DEQUEUE Coeffs
06      UNTIL Coeffs IS EMPTY
07      RETURN Answer
08  ENDFUNCTION
```

Solution 2: Add the last item in the queue without multiplying by X:

```
01  FUNCTION Evaluate(X : INTEGER, Coeffs : QUEUE) RETURNS INTEGER
02      Answer ← 0
03      REPEAT
04          Answer ← Answer * X
05          Answer ← Answer + DEQUEUE Coeffs
06      UNTIL LENGTH(Coeffs) = 1
07      Asnwer ← Answer + DEQUEUE Coeffs
08      RETURN Answer
09  ENDFUNCTION
```

4(a) Possible answers

- Privacy of students and other users

- No price gouging

- No retaining of unnecessary data

4(b)

- Data is divided into packets, each with its own ID number

- Each packet is sent from one node to another on the internet.

- At each point, each node decides where to forward the packet based on network traffic and node availability.

- Nodes are received and reassembled in order using the ID number

4(c)(i)

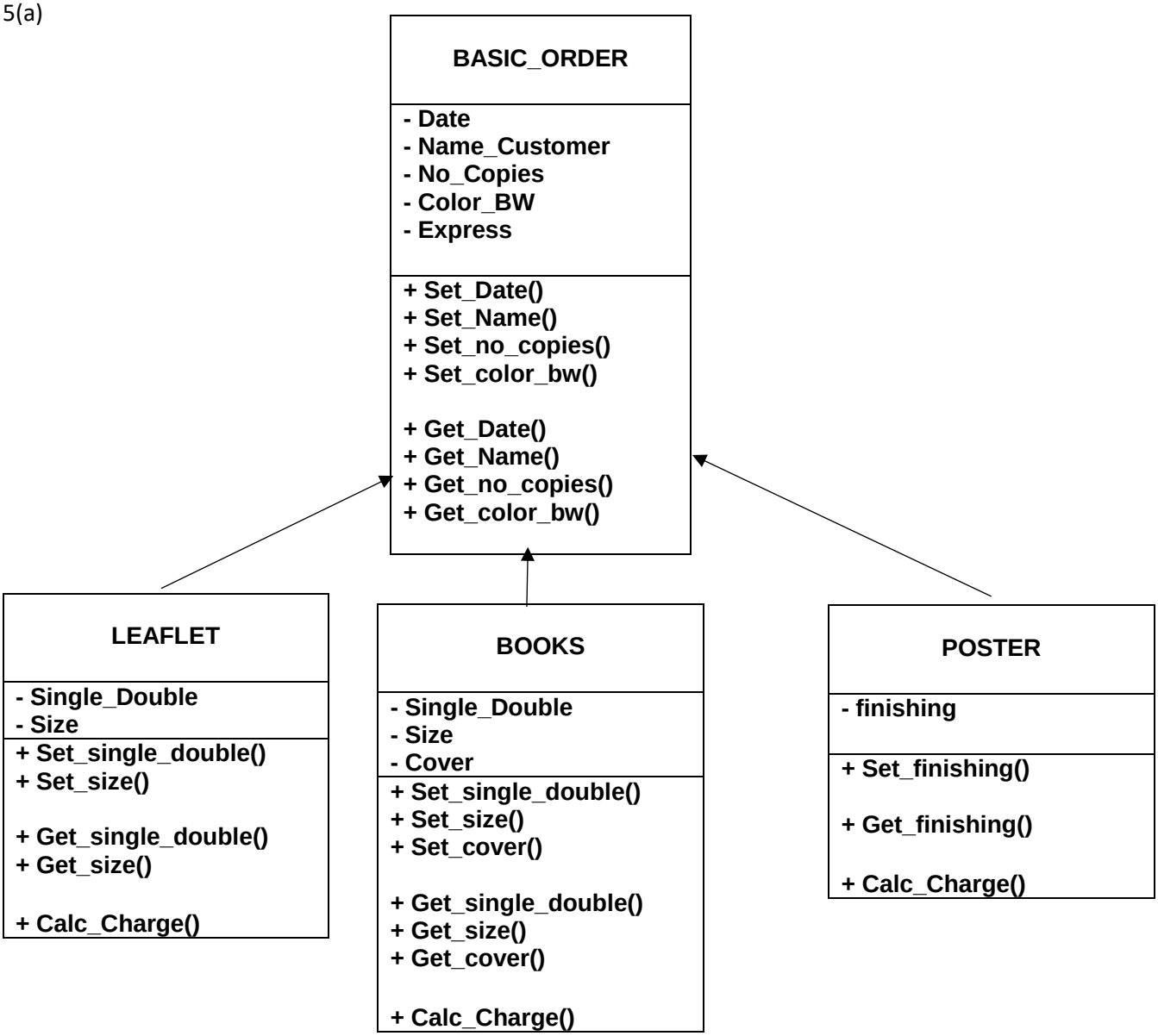
- Problem is due to different encoding and decoding systems being used to input/output non-ASCII characters

- Communication protocols needed to ensure that browser uses correct decoding to read the webpage

4(c)(ii)

- Unicode intended to be universal standard across all languages and systems so that there is no need to toggle between different decoding systems for different languages

5(a)

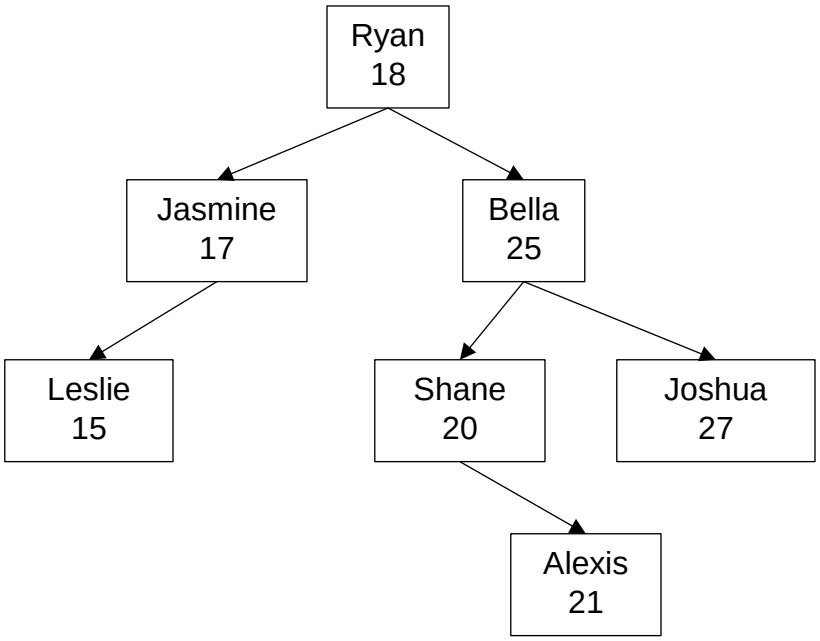


- 1 m: base class + 1 derived class
- 1m: remaining 2 derived classes
- 1m: properties of base class and 1 derived class
- 1m: methods of base class and 1 derived class
- 1m: properties remaining 2 derived class
- 1m: methods of remaining 2 derived class
- 1m: suitable inheritance from base class / among derived classes
- 1m: method to calculate charge in all derived classes

5(b)

- Inheritance allows reusability and
- thereby reducing the time needed for implementation. / improve maintainability of code

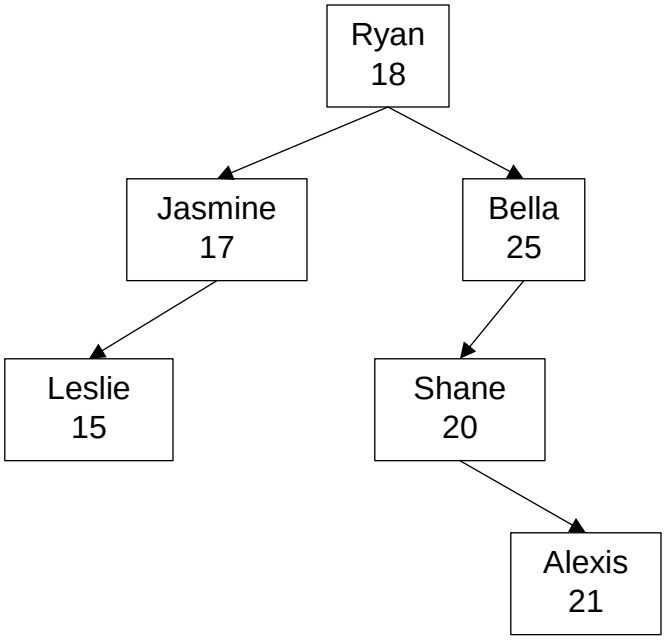
6(a)



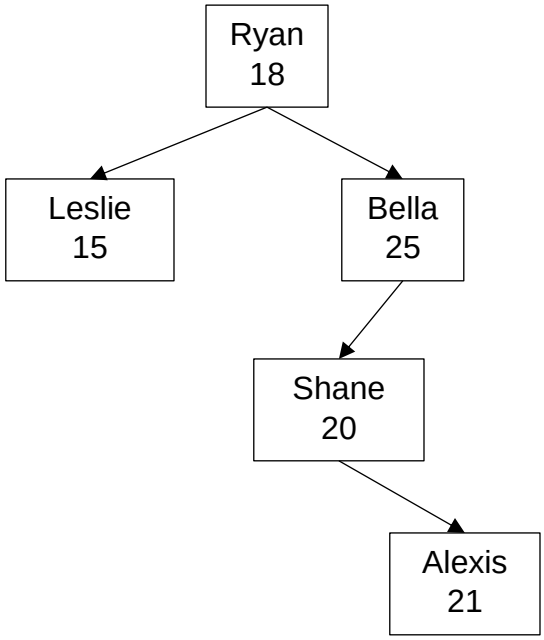
6(b)

Index	Name	Score	Left Pointer	Right Pointer
0	Ryan	18	4	1
1	Bella	25	3	2
2	Joshua	27	None	None
3	Shane	20	None	5
4	Jasmine	17	6	None
5	Alexis	21	None	None
6	Leslie	15	None	None

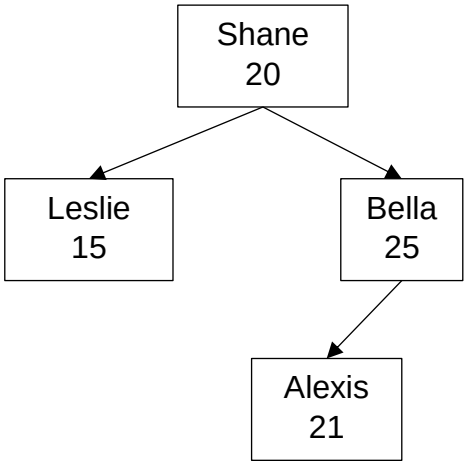
6(c)(i)


Ryan
18
Jasmine
17
Bella
25
Leslie
15
Shane
20
Alexis
21

6(c)(ii)


Ryan
18
Leslie
15
Bella
25
Shane
20
Alexis
21

6(c)(iii)



6(d)

Recursive program.

If the root's score is greater than input value, add the root to the list, then add every node in the right subtree to the list. Run the program recursively on the left subtree.

If the root's score is less than or equal to input value, (ignore left subtree and) run the program recursively on the right subtree.

7(a)(i)

A: j ← j + 1

B: MyList[i] ← Temp

j ← j + 1

i ← i + 1

C: RETURN I

D: R-L >= 1

E: CALL Quicksort(PivotPos + 1, R, MyList)

7(a)(ii)

$O(n^2)$

7(a)(iii)

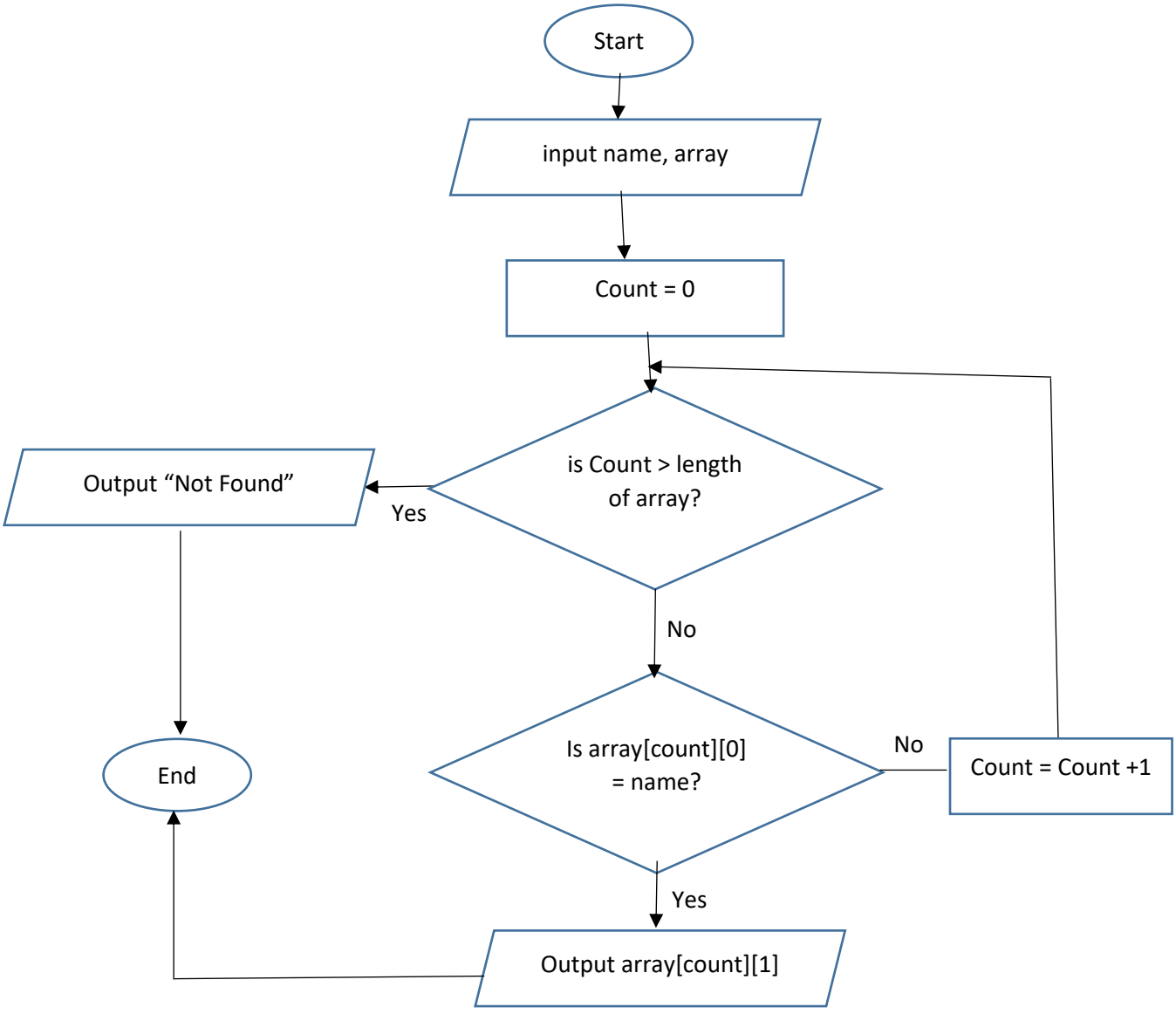
- list is already sorted in ascending order
- The partition function would perform swaps for every comparison and

- Consistently return the end of the list as the pivot. This leads to the maximum number of recursive calls required and a time complexity of $O(n^2)$

7(a)(iv)

- For a list that is already sorted, insertion sort does not need to make any swaps,
- and would only need to make n comparisons through one iteration, leading to a time complexity of O(n).
  Hence insertion sort is more efficient.

7(b)(i)



7b(ii)

- Hash table has O(1) for checking presence of a name, which is more efficient than an array's O(lg(n))
- An array can be sorted and remain sorted but not for a hash table.

7b(iii)

- Initialise an array that is much larger than the expected number of items to be stored.
- Hash the name with a hashing algorithm.
- Assign the name and score to the array index calculated by the hash function.

8(a)

Name: string

Class: string

Score: float

8(b)(i)

SQL	NoSQL
Has fixed, predefined schema	No predefined schema, thus dynamic and can change easily
Data are stored in tables with a fixed data type in each field	Data are stored as collections of documents with no fixed data types
Joins are used to get data across tables, thus easier to use for complex queries	No join operations

8(b)(ii)

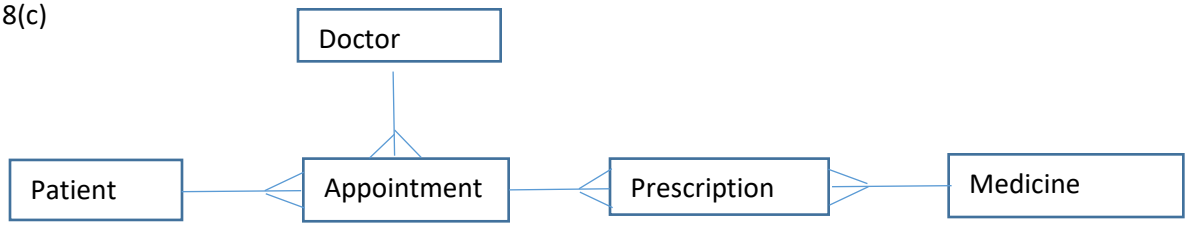
Possible answer in SQL's favour:

- Since the data stored has a fixed schema and flexibility afforded by NOSQL is unnecessary,
- Complex queries involving join operations are required to retrieve data across multiple tables
- ACID compliance is important
- There is possibly a high number of simultaneous transactions performed during mark entry period

Answer in NOSQL's favour:

- NoSQL databases can take advantage of multiple servers, ensuring the application functions even if one server fails
- NoSQL provides flexibility in adding/removing data fields should the requirements of the application change with time
- Horizontally scalable, hence schools might find it easier to add an additional low cost server if the need arises, as compared to replacing existing server with a better one

8(c)



8(d)

Patient (Pa_NRIC, Name)

Doctor (Dr_NRIC, Name)

Appointment (Date, Time, Pa_NRIC*, Dr_NRIC*, Presc_ID)

Prescription (Presc_ID*, Med_ID*)

Medicine (Med_ID, Name, Price)

5 marks: 1 m each for primary keys of each table

1m: 3 foreign keys in appointment and prescription

Note: Presc_ID functions like an Appt_ID, it is unique for every appointment (question assumes all patients will be given a prescription after the appointment)