Sri Lanka Institute of Information Technology

B.Sc. Honours Degree in Information Technology

Specialized in Information Technology

Final Examination
Year 2, Semester 2 (2025)

IT2070 – Data Structures and Algorithms

Duration: 2 Hours

June 2025

Instructions to Candidates:

- This paper has 4 questions.
- Exam commences with 10 minutes reading period
- Answer all questions in the booklet given.
- The total marks for the paper is 100.
- This paper contains 7 pages, including the cover page.
- Electronic devices capable of storing and retrieving text, including calculators and mobile phones are not allowed.

Question 1 **25 marks**

(1) "A stack can be created using a linked list". Do you agree with this statement? Briefly discuss your opinion. (3 marks)

Agree → 1 mark

The 2 main operations of the stack namely push () and pop() methods can be implemented by insertFirst() and deleteFirst() methods in linkedlist respectively. →2 marks

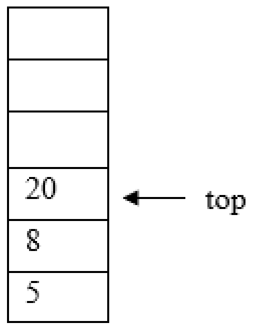
(2) "The performance of a circular queue is greater than a linear queue". Briefly explain your opinion about this statement. (2 marks)

In a circular queue there is no limitation of front and rear indexes (they can move anywhere). But, in linear queue, the limitation exists. Therefore, more operations can be processed within a circular queue than linear queue (memory wastage is minimum in circular queue). So, performance is high in circular queue.

(3) How does a queue can be used to store the keystroke data as a user type at the Keyboard? (2 marks)

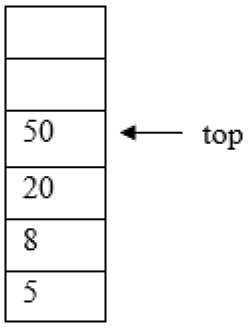
The same order that a user types at the keyboard should be maintained when they are storing. So, FIFO order should be followed. Therefore, a queue should be used to support that functionality.

(4) Consider the following stack data structure. Draw the stack frames after executing the following operations and indicate the values of top and count. (6 marks)



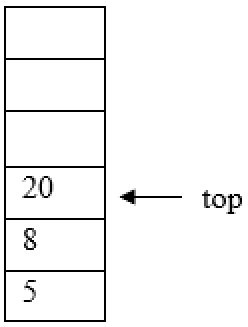
(a) push(50)
(b) pop()
(c) peek()
(d) pop()
(e) pop()
(f) pop()

push(50)



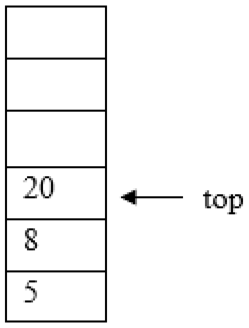
top=3 count=4

pop()



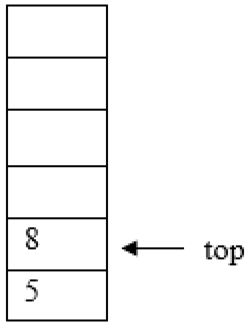
top=2 count=3

peek()



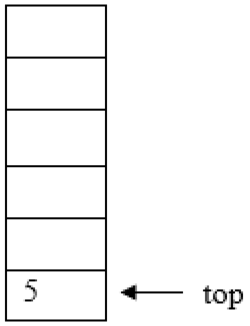
top=2 count=3

pop()



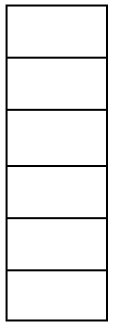
top=1 count=2

pop()



top=0 count=1

pop()



top= -1 count=0

1 mark for each

(5) If the same operations have been done for a linear queue with the size of 6 with the same content, state the values of front, rear and count for each operation. (3 marks)

Note that the push(), pop() and peek() operations of a stack can be aligned with insert(), remove() and peekFront() methods in a queue. Consider front points to element 5 and rear points to element 20.

push(50) → front = 0	rear = 3	count=4
pop()→ front = 1	rear = 3	count=3
peek()→ front = 1	rear = 3	count=3
pop()→ front = 2	rear = 3	count=2
pop()→ front = 3	rear = 3	count=1
pop()→ front = 4	rear = 3	count=0

0.5 marks for each

(6) Write a main program in Java to remove all the occurrences of a given element of a stack. (9 marks)

Note that the size of the stack, content and the element to be removed should be taken as keyboard inputs.

Assume that the StackX class has been already implemented with push(), poop(), peek(), isEmpty() and isFull() methods.

Input :

Stack elements:

10 4 10 5 10

Element to be removed:

10

Output:

4 5

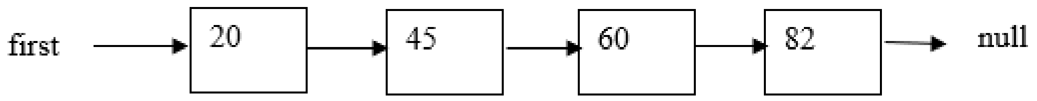
```
import java.util.Scanner;

public class StackMain {
        public static void main(String [] args)
        {
                int size1;
                int val, temp;

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the size of the stack:");
        size1 = sc.nextInt(); → 1 mark

        StackX s1= new StackX(size1);  →          1 mark
        StackX s2= new StackX(size1);

        System.out.println("Enter the elements of the stack:");

        for(int i=0; i<size1; i++) → 1 mark
                s1.push(sc.nextInt()); → 1 mark

        System.out.print("Enter the value to be removed:");
        val = sc.nextInt();

        while(!s1.isEmpty()) → 1 mark
        {
                temp = s1.pop(); →1 mark
                if(temp!=val) → 1 mark
                        s2.push(temp); → 1 mark
        }

        while(!s2.isEmpty())
                System.out.println(s2.pop()); → 1 mark

        }

}
```

Question 2 **25 marks**

(1) Compare and contrast array and a linked list. (2 marks)

Array	**Linked List**
A data structure	A data structure
Direct access is possible	Access should be started from first always

(2) Consider the linked list given below and state the output generated after executing each of the code segments. (6 marks)



(a)

```
Link current = first;
while (current != null)
{
        System.out.print (current.iData + " ");
        current = current. next;
}
```

Output → 20,45,60,82 → 2 marks

(b)

```
Link current = first;
current = current. next;
while (current != null)
{
        System.out.print (current.iData + " ");
        current = current. next;
}
```

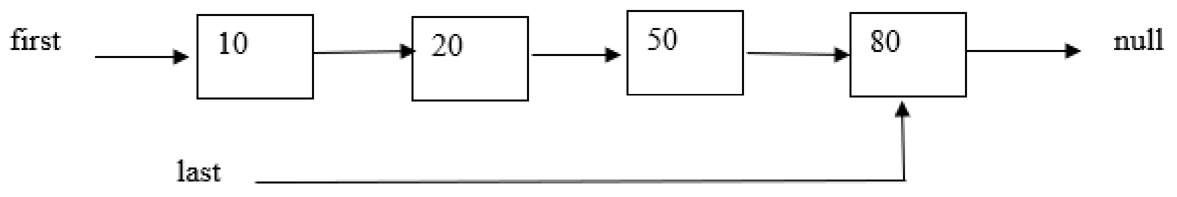
Output →45,60,82 → 2 marks

(c)

```
Link current = first;
while (current.next != null)
{
        System.out.print (current.iData + " ");
        current = current. next;
}
```

Output →20,45,60 → 2 marks

(3) Consider the following double-ended list. Write down the code segments for the following instances. (8 marks)



(a) Delete link 50

first.next.next = last; → 2 marks

(b) Delete and return 10

Link temp = first; → 1 mark

first = first.next; → 1 mark

return temp; → 1 mark

(c) Insert 30 after 10

Link nl = new Link(30); → 1 mark

nl.next = first.next; → 1 mark

first.next = nl; → 1 mark

(4) Compute the number of nodes of full binary tree with height 3. (1 mark)

$n = 2^{h+1} - 1$ → 0.5 marks

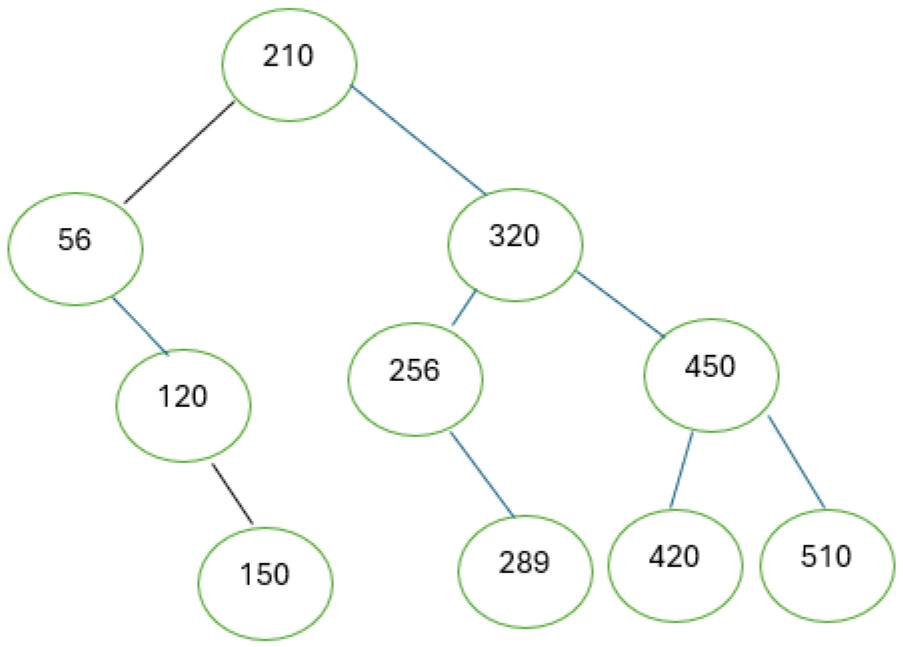
$n = 2^{3+1} - 1$

$n = 15$ → 0.5 marks

(5) Consider the following set of nodes.

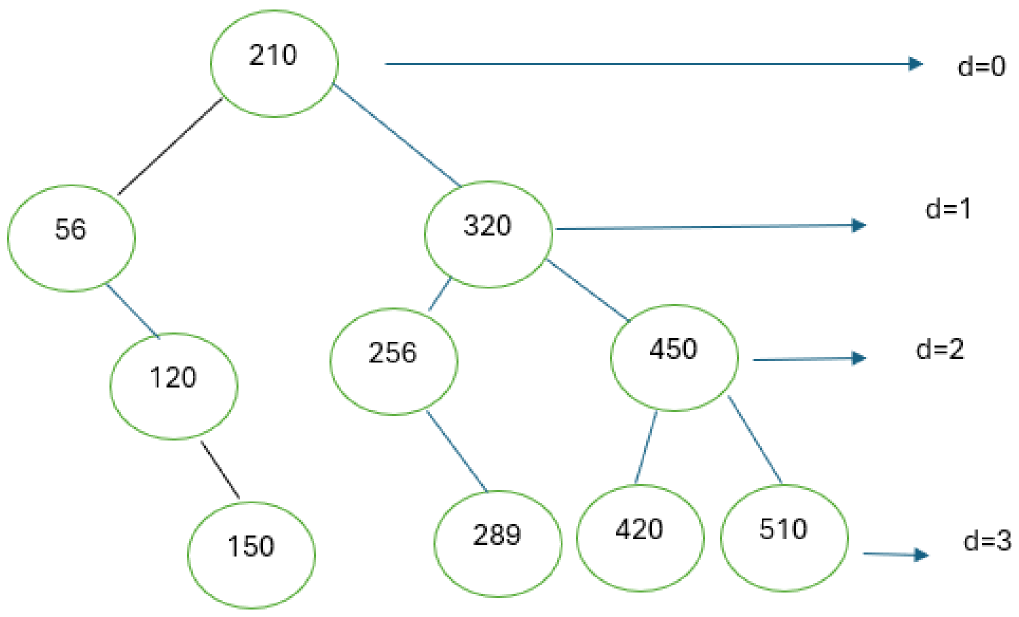
210,56,120,320,450,150,256,510,289,420

(a) Arrange these nodes into a binary search tree and draw that tree. (5 marks)



0.5 marks for each node

(b) Indicate the depth levels in the tree that you have drawn in (5) (a). (2 marks)



0.5 marks for each depth level

(c) What is the height of this tree? (1 mark)

Height = max depth

= 3

Question 3 **25 marks**

(1) State whether the following statements are either TRUE OR FALSE. (5 marks)

(a) Asymptotic notations are the most practical way of representing the complexity of an algorithm. → TRUE
(b) Complexities of Heapsort and Mergesort algorithms are same. → TRUE
(c) Complexities of Insert() and Extract() of a priority queue are same. → TRUE
(d) Divide and Conquer Method is an algorithm designing technique. → TRUE
(e) Mergesort is a not recursive algorithm. → FALSE

1 mark for each

(2) Express the T(n) of following pseudocodes using RAM model. (6 marks)

(a)

```
a = 5
while a > 0
        a = a - 2
        print a
```

(b)

```
for i = 1 to 5
        for j = 1 to n-1
                a = A[i] + i * j
                print a
```

```
a = 5 --> 1
while a > 0 --> 4
        a = a - 2 --> 6
        print a --> 3

total = 14
```

2 marks for the breakdown and 1 mark for the total

```
for i = 1 to 5 --> 17
        for j = 1 to n-1 -->5(3n-1)
                a = A[i] + i * j-->5(4(n-1))
                print a --> 5(n-1)

        total = 40n - 13
```

2 marks for the breakdown and 1 mark for the total

(3) "The Complexities of Quick Sort best case and Merge Sort are equal". Do you agree with this statement? Briefly explain the reason. (2 marks)

Yes

Quick Sort best case and Merge Sort has balanced partitions. So, their complexities are equal, which is $O(n\log_2 n)$.

(4) What is the recurrence equation for the Merge Sort algorithm? (2 marks)

$$T(n) = 2T(n/2) + c.n$$

(5) Solve the above (4) recurrence equation using Master Theorem. (3 marks)

$$T(n)=\begin{cases}\Theta\left(n^{\log_b a}\right) & f(n)=O\left(n^{\log_b a-\varepsilon}\right)\rightarrow f(n)<n^{\log_b a}\\ \Theta\left(n^{\log_b a}\lg n\right) & f(n)=\Theta\left(n^{\log_b a}\right)\ \rightarrow f(n)=n^{\log_b a}\\ \Theta\left(f(n)\right) & f(n)=\Omega\left(n^{\log_b a+\varepsilon}\right)\ \rightarrow f(n)>n^{\log_b a}\\ & \text{if } af(n/b)\le cf(n) \text{ for } c<1 \text{ and large } n\end{cases}$$

T(n) = 2T(n/2) + c.n

a = 2 b = 2 f(n) = c.n → 1 mark

f(n) vs n^$^{\text{log b a}}$

c.n vs n^log $_2$ 2

n vs n → case 2 → 1 mark

T(n) = O(log $_2$ n) → 1 mark

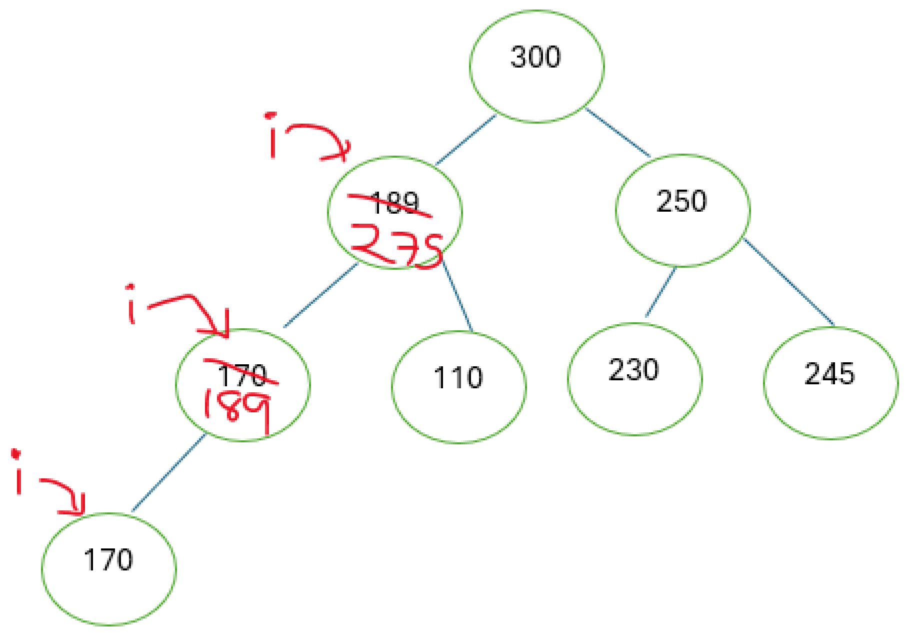
(6) Illustrate the operations of Heap_Insert(A, 275) for heap A with given set of elements. (For illustration process assign the values only once and then use a diagrammatic approach to reach the answer.) (7 marks)

A

300	189	250	170	110	230	245

```
HEAP_INSERT(A, key)
1 A.heap_size = A.heap_size + 1
2 i = A.heap_size
3 while i > 1 and A[PARENT(i)] < key
4       A[i] = A[PARENT(i)]
5       i = PARENT(i)
6 A[i] = key
```

```
HEAP_INSERT(A, key)
1 A.heap_size = A.heap_size + 1 = 8 → 1 mark
2 i = A.heap_size = 8
3 while 8 > 1 and 170 < 275 → 1 mark
4       A[8] = 170 → 1 mark
5       i = 4 → 1 mark
6 A[2] = 275 → 1 mark
```



2 marks for the diagram

Question 4 **25 marks**

(1) Using suitable example, derive the time complexity of the best case of Naïve String Matching algorithm. (4 marks)

Any suitable example with a Pattern (first letter does not exist inside the Text) → 2 marks and derives the complexity as O(n-m+1)→ 2 marks

(2) Compute the number of valid and invalid shifts for the following Text and the Pattern with the use of Naïve String Matching algorithm. (4 marks)

T = 105680760
P = 50

	1	0	5	6	8	0	7	6	0	Number of comparisons
S=0	5	0								1
S=1		5	0							1
S=2			5	0						2
S=3				5	0					1
S=4					5	0				1
S=5						5	0			1
S=6							5	0		1
S=7								5	0	1

2 marks for the table

Valid shifts = 0 → 1 mark
Invalid shifts = 8 → 1 mark

(3) Derive the number of valid and spurious hits if you apply the same Text and the Pattern in (2) with Rabin Carp algorithm with the modulo value (q) = 10. (5 marks)

T = 105680760

P = 50
q = 10

P % q = 50 % 10 = 0 → 1 mark

10% 10= 0 → spurious hit

05% 10=5

56% 10=6

68% 10=8

80% 10=0 → spurious hit

07% 10=7

76% 10=6

60% 10=0 → spurious hit

2 marks for the combinations

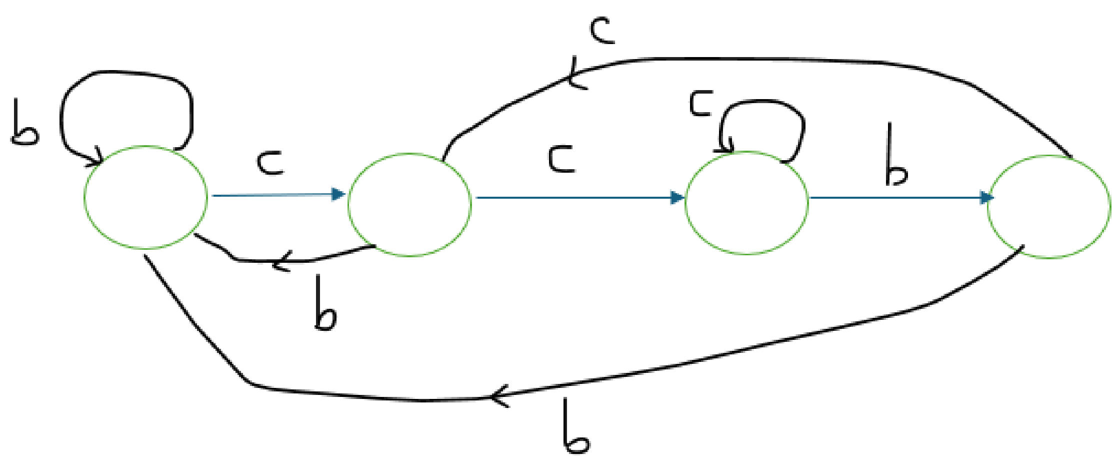
Valid hits = 0 → 1 mark

Spurious hits = 3 → 1 mark

(4) Which computation is better comparing to Naïve String Matching and Rabin Carp algorithms? Briefly explain your answer. (3 marks)

Rabin Carp Algorithm → 1 mark
The number of comparisons in Rabin Carp algorithm is less than Naïve String Matching algorithm. → 2 marks

(5) Draw the state transition diagram for Pattern P = ***ccb*** along with the input alphabet = {b,c}. (9 marks)



Identify 4 states → 1 mark
Each transition → 1 mark (8*1=8 marks)

-- ***End of the Paper***---