UIU QUESTION BANK

FINAL QUESTION SOLUTIONS

DATA STRUCTURE AND ALGORITHM I

CSE 2215

SOLUTION BY

NURUL ALAM ADOR

*UPDATED TILL **FALL 2023***

nurulalamador.github.io/**UIUQuestionBank**

Index

Fall 2023

1. a) Which Data Structures are appropriate to implement the following and why?

i) Matrix representation
ii) Bus Ticket Counter
iii) Different locations of Dhaka City with a distance from one another

Solution:

i) Two Dimensional Array is appropriate Data Structure to represent Matrix. Because in Two Dimensional Array, we can store our data row and column wise like Matrix.

ii) Queue is appropriate Data Structure to implement Bus Ticket Counter. Because in Bus Ticket Counter lines, who came first he get ticket first, which is similar to Queue's First-In-First-Out policy.

iii) Different locations of Dhaka City with a distance from one another can be representing with Weighted Graph. The vertices will represent the locations name and edges weight will be represent the distance.

1. b) Convert the following infix expression into postfix using a STACK.

Infix expression: a↑2-(b+c/d*a+b)

Solution:

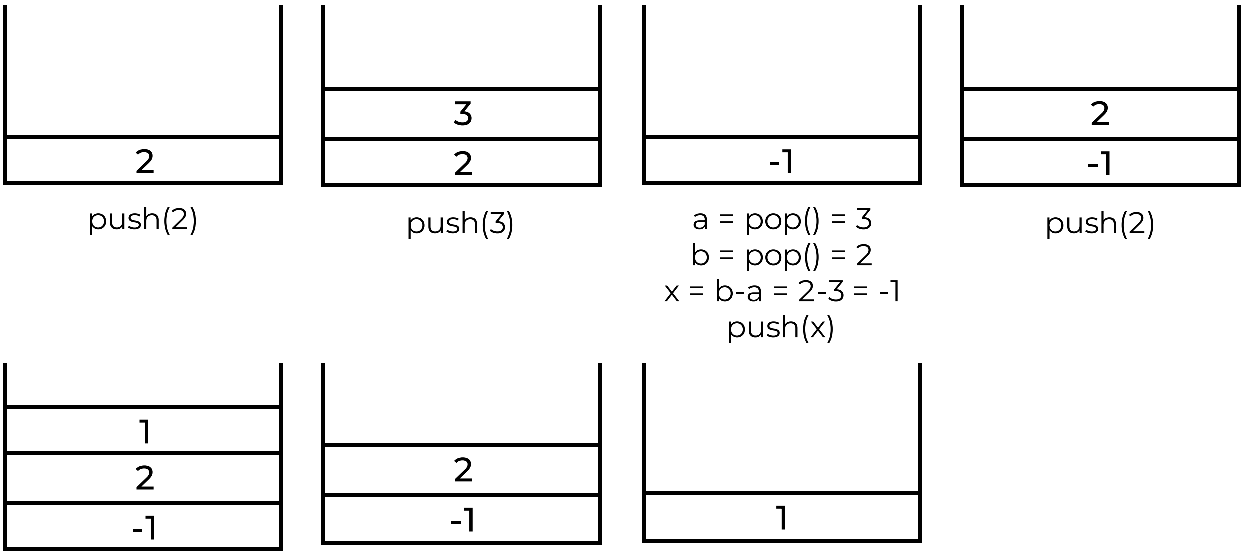
Symbol	Stack	Postfix Expression
a		a
↑	↑	a
2	↑	a2
-	-	a2↑
(	-(	a2↑
b	-(	a2↑b
+	-(+	a2↑b
c	-(+s6	a2↑bc
/	-(+/	a2↑bc
d	-(+/	a2↑bcd
*	-(+*	a2↑bcd/
a	-(+*	a2↑bcd/a
+	-(+	a2↑bcd/a*+
b	-(+	a2↑bcd/a*+
)	-	a2↑bcd/a*+b+
		a2↑bcd/a*+b+-

∴ Final Postfix Expression: a2↑bcd/a*+b+-

1. c) Evaluate the postfix expression, a b – c d * + for a=2, b=3, c=2 and d=1.

Solution:

2 3 – 2 1 * +



push(2)

push(3)

a = pop() = 3
b = pop() = 2
x = b-a = 2-3 = -1
push(x)

push(2)

push(1)

a = pop() = 1
b = pop() = 2
x = b*a = 2*1 = 2
push(x)

a = pop() = 2
b = pop() = -1
x = b+a = -1+2 = 1
push(x)

∴ Final result is 1.

1. d) Design an iterative algorithm for TOWER OF HANOI using one intermediate pillar/peg and show simulation for n = 2, where n is the number of disks

Solution:

Algorithm:

```
class Hanoi {
    int n;
    int s;
    int a;
    int d;
}

int towerOfHanoi(int n, int s, int d, int a) {
    Stack stack = new Stack();
    stack.push(new Hanoi(n,s,d,a));

    while(stack.isEmpty() == false) {
        Hanoi temp = stack.pop();
        n=temp.n, s=temp.s, d=temp.d, a=temp.a;

        if(n==1) {
            printf("%d -> %d", s, d)
        } else {
            stack.push(new Hanoi(n-1,s,a,d));
            stack.push(new Hanoi(1,s,d,a));
            stack.push(new Hanoi(n-1,a,d,s));
        }
    }
}
```

Simulation:

Hanoi (2, 1, 3, 2)

Sequence:

Hanoi (1, 1, 2, 3)
Hanoi (1, 1, 3, 2)
Hanoi (1, 2, 3, 1)
~~Hanoi (2, 1, 3, 2)~~

Sequence:

~~Hanoi (1, 1, 2, 3)~~
Hanoi (1, 1, 3, 2)
Hanoi (1, 2, 3, 1)
~~Hanoi (2, 1, 3, 2)~~

Sequence:

1 → 2

~~Hanoi (1, 1, 2, 3)~~
~~Hanoi (1, 1, 3, 2)~~
Hanoi (1, 2, 3, 1)
~~Hanoi (2, 1, 3, 2)~~

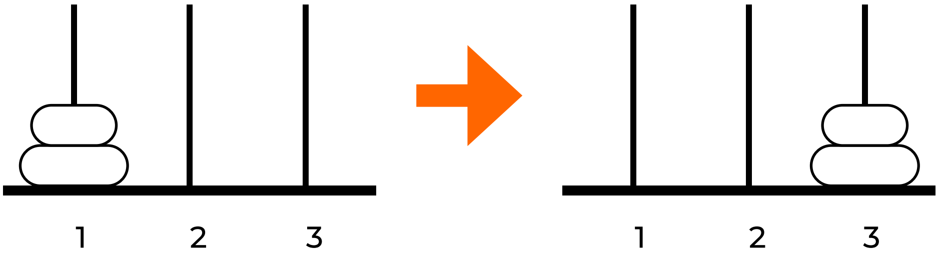
Sequence:

1 → 2
1 → 3

~~Hanoi (1, 1, 2, 3)~~
~~Hanoi (1, 1, 3, 2)~~
~~Hanoi (1, 2, 3, 1)~~
~~Hanoi (2, 1, 3, 2)~~

Sequence:

1 → 2
1 → 3
2 → 3



2. a) Show the manual tracing of the following algorithm using given Queue of size 3. Here, Queue is FIFO data structure, and m, f and r are size, front and rear of the Queue, respectively. What is the purpose of the algorithm?

Queue	32	30	[redacted]	31
	0	r=1	f=2	3

```
i=(f+1)%(m+1);
while(i!=f) {
    printf("%d ", Queue[i]);
    i=(i+1)%(m+1);
}
```

Solution:

Here, f = 2, r = 1, m = 3
∴ i = (2+1)%(3+1) = 3%4 = 3

i	i != f	Output	i = (i+1)%(m+1)
3	3 != 2 (True)	Queue[3] = 31	i = 3+1%3+1 = 4%4 = 0
0	0 != 2 (True)	Queue[0] = 32	i = 0+1%3+1 = 1%4 = 1
1	1 != 2 (True)	Queue[1] = 30	i = 1+1%3+1 = 2%4 = 2
2	2 != 2 (False)		

∴ Final Output: 31 32 30

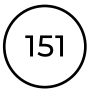
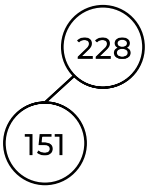
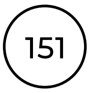
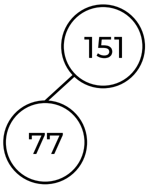
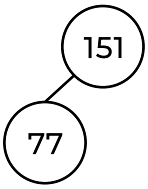
By considering manual tracing, we can say the purpose of this algorithm is printing all element of the Queue except the front element, which is blank.

2. b) Show the status of a QUEUE and a Priority QUEUE (Data in Descending Order) for the following operations, where both QUEUEs are implemented by an array of size m=3. Here, Enqueue and Dequeue mean insert and delete respectively, and x= last two digits of your student id+4, y=x+3, z=x+y and p=y+z.

Enqueue(z), Enqueue(p), Dequeue(), Enqueue(y), Enqueue(z)

Solution:

Here, x = 70+4 = 74　　　z = 74+77 = 151
y = 74+3 = 77　　　p = 77+151 = 228

	Queue	Priority Queue
Enqueue(z)	151	
Enqueue(p)	151 228	
Dequeue()	228	
Enqueue(y)	228 77	
Enqueue(z)	228 77 151	 [Duplicate elements normally can't be inserted in Priority Queue]

2. c) Find a min-heap tree from the following data where ID=last two digits of your student ID. Show step by step procedure during the construction of the tree.

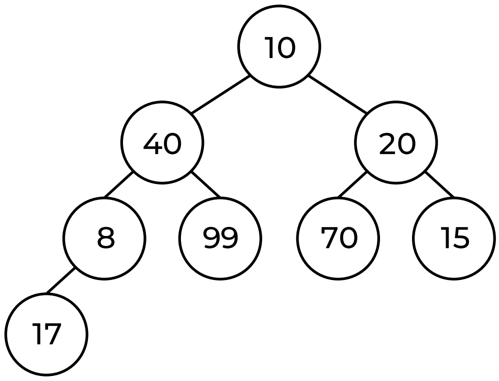
10 40 20 8 99 ID 15 17

Solution:

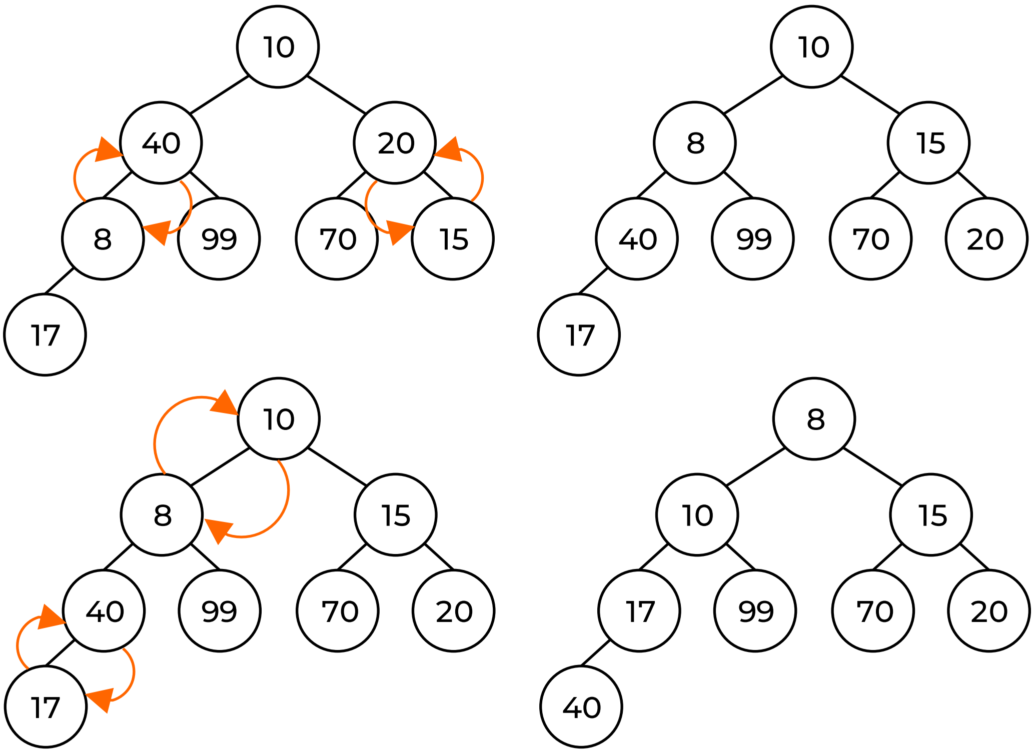
Here,

10 40 20 8 99 70 15 17

Constructing Complete Binary Tree:



Constructing Min-Heap:

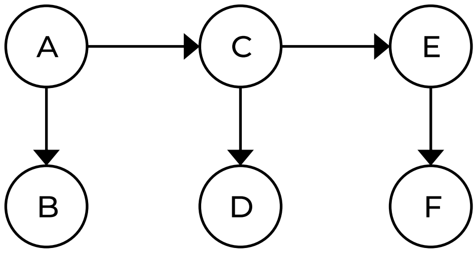


Our Min-Heap has been constructed successfully.

3. a) Draw a directed acyclic graph with six vertices

Solution:

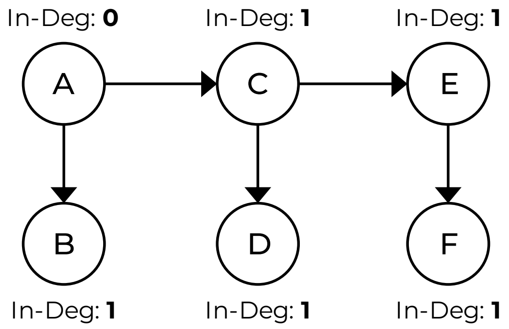
The acyclic graph has been drawn below:



3. b) Show the simulation of Topological Ordering Algorithm using graph in Ques. 3(a).

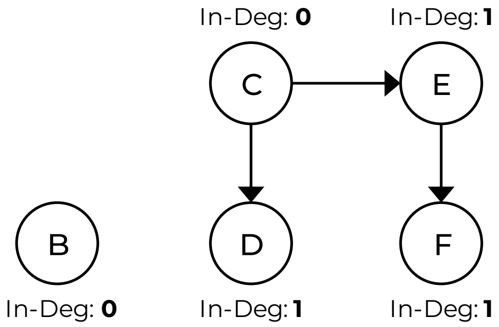
Solution:

Simulation of Topological Ordering Algorithm has been showed below:



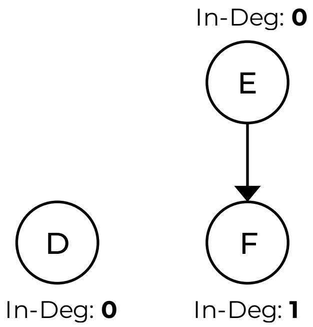
<u>Topological Order:</u>

A



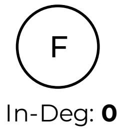
<u>Topological Order:</u>

A, B, C



<u>Topological Order:</u>

A, B, C, D, E



<u>Topological Order:</u>

A, B, C, D, E, F

∴ Final Topological Order: A, B, C, D, E, F.

3. c) Draw a binary tree using the data given below, where x, y, z, p, r, t, u and v are nodes of the tree.

y p z x r t u v

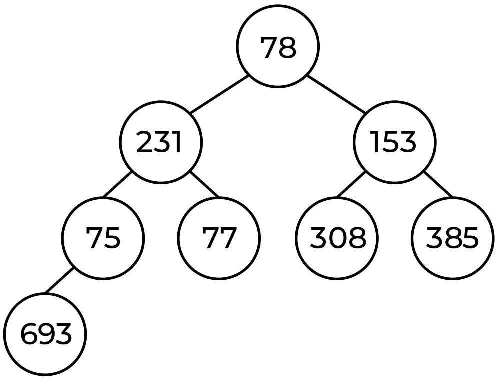
Here, x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r, u=r+t, v=t+u

Solution:

Here, x = 70+5 = 75
y = 75+3 = 78
z = 75+78 = 153
p = 78+153 = 231
r = 75+2 = 77
t = 231+77 = 308
u = 77+308 = 385
v = 308+385 = 693

78	231	153	75	77	308	385	693
y	p	z	x	r	t	u	v

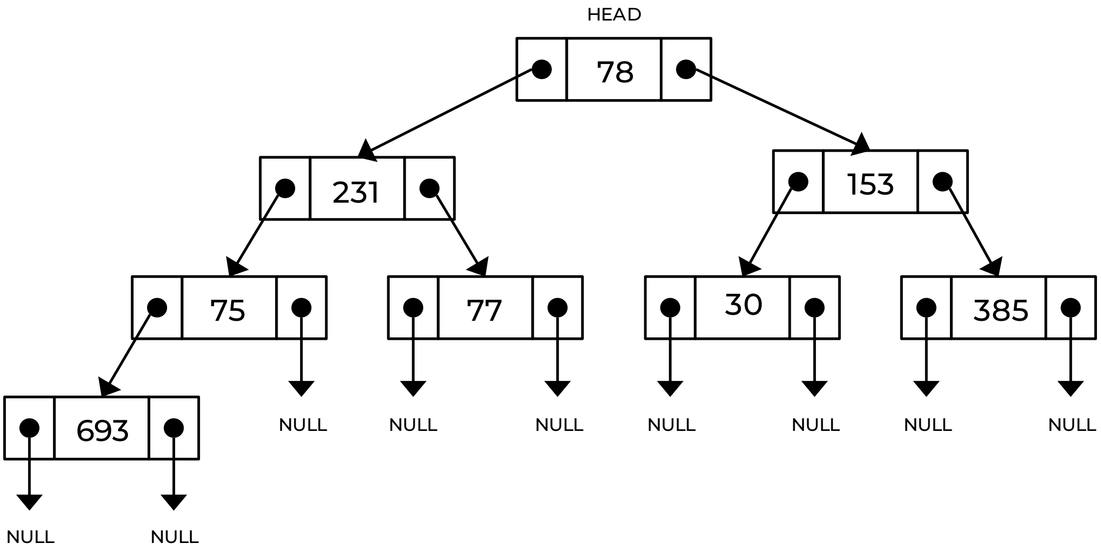
Now, the binary tree has been drawn below:



3. d) How can you represent the binary tree of Ques. 3(c) using one dimensional array and linked list?

Solution:

Linked List Representation:



One Dimensional Array Representation:

78	231	153	75	77	308	385	693
0	1	2	3	4	5	6	7

Here,
Left Child = i*2+1
Right Child = i*2+2
Parent = ceil(i/2)-1

4. a) Draw a binary tree from the following Inorder and Postorder sequences

Inorder: v p y r x t z u
Postorder: v p r y t u z x

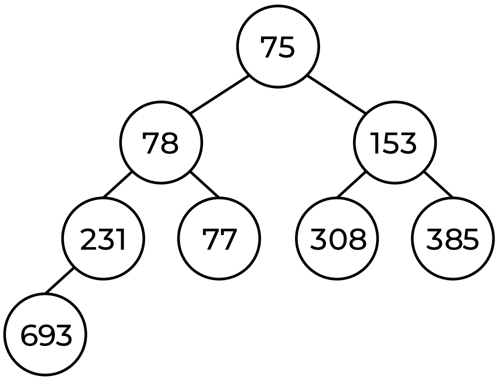
Here, x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r, u=r+t, v=t+u

Solution:

Here,
x = 70+5 = 75
y = 75+3 = 78
z = 75+78 = 153
p = 78+153 = 231
r = 75+2 = 77
t = 231+77 = 308
u = 77+308 = 385
v = 308+385 = 693

Inorder:	693	231	78	77	75	308	153	385
Postorder:	693	231	77	78	308	385	153	75

The binary tree has been drawn below:



4. b) Construct a binary search tree (BST) using the nodes y, p, z, x, r and t, where x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r. Show the insertion and deletion of r+t and y, respectively in/from the BST.

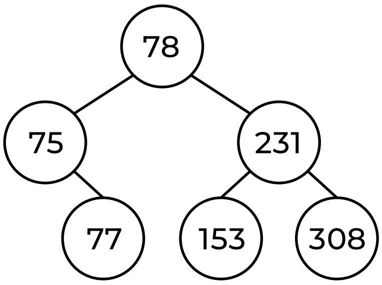
Solution:

Here,
x = 70+5 = 75
y = 75+3 = 78
z = 75+78 = 153
p = 78+153 = 231
r = 75+2 = 77
t = 231+77 = 308

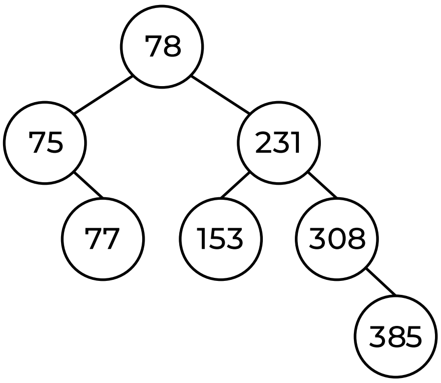
78	231	153	75	77	308
y	p	z	x	r	t

The binary search tree has been drawn below:

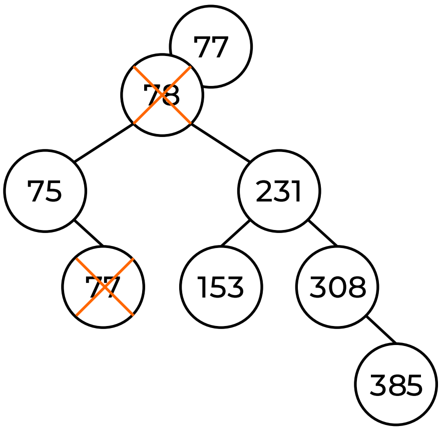
[P.T.O]

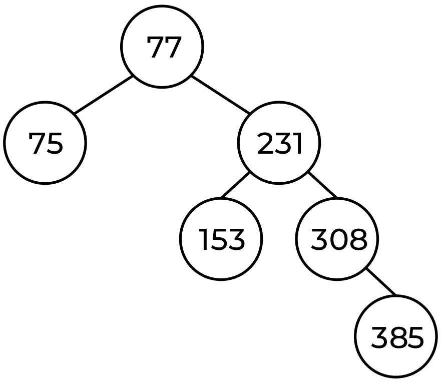


Inserting r+t: *(77+308 = 385)*



Deleting y: *(78)*





4. c) Two disjoint sets {y, p, z, x} and {r, t} are given where minimum one of a set is the representative of that set. Determine UNION(Find(x), Find(t)). How can you check x and y are in the same set using Find operation? Here, x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r.

Solution:

Here,	x = 70+5 = 75	p = 78+153 = 231
	y = 75+3 = 78	r = 75+2 = 77
	z = 75+78 = 153	t = 231+77 = 308

{ 78, 231, 153, 75 }
Minimum: 75

{ 77, 308 }
Minimum: 77

Representing these disjoint sets with array:

[P.T.O]

	75		77		75		75		75		77	
...	75	...	77	...	78	...	153	...	231	...	308	...

Now,
UNION(Find(x), Find(t))

Here,
Find(x) = Find(75) → 75
Find(t) = Find(308) → 77

After executing UNION(75, 77):

	75		75		75		75		75		75	
...	75	...	77	...	78	...	153	...	231	...	308	...

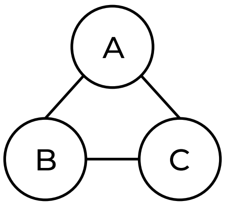
Again,
Find(x) = Find(75) → 75
Find(y) = Find(78) → 75

Since Find(x) and Find(y) are returning same value, therefore they are in same set.

4. d) Develop an idea to check whether addition of an edge in a binary tree forms a cycle or not using Find operation.

Solution:

To check there are any cycle or not, at first we need to make different disjoint sets for every vertex. Then we should check connecting vertices of every edge, are they are from different set or not by using Find operation. If they are not in same set, we should Union both vertices disjoint set. While running this process, if we find any edge which connected vertices are from same set, then there is cycle in our structure and that is a cyclic graph, not a binary tree. If we don't find any edge like this, that's will mean there are no cycle in our binary tree. A simulation of this process are shown below:



Let,
S1 = { A }
S2 = { B }
S3 = { C }

<u>For Edge A-B:</u>
Find(A) = S1
Find(B) = S2
S1 ≠ S2
Union(S1, S2) → S4 = { A, B }

<u>For Edge A-C:</u>
Find(A) = S4
Find(C) = S3
S4 ≠ S3
Union(S4, S3) → S5 = { A, B, C }

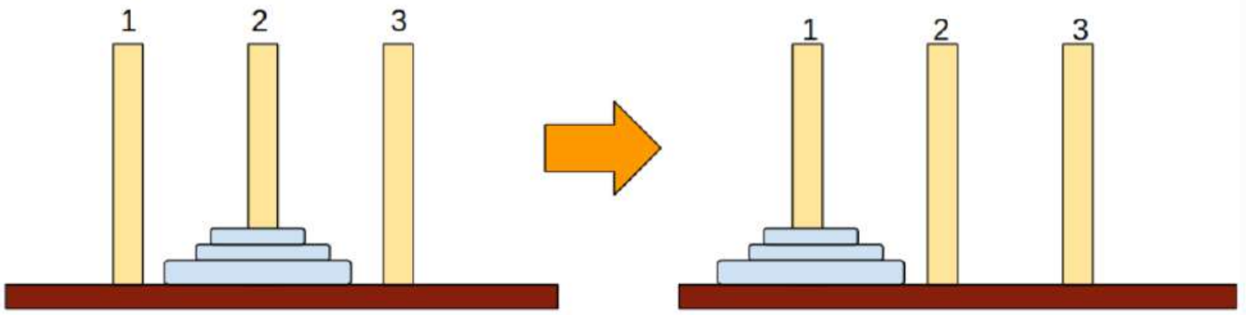
<u>For Edge B-C:</u>
Find(B) = S5
Find(C) = S5
S5 == S5

Since, Find(B) and Find(C) returning same set,
∴ There is cycle in our graph.

Summer 2023

1. a) Consider a recursive function `int Hanoi(n,s,d,a)`.
Here:
`n` – represents the number of discs on the source peg.
`s` – denotes the source peg number.
`d` – denotes the destination peg number.
`a` – denotes the auxiliary peg number.

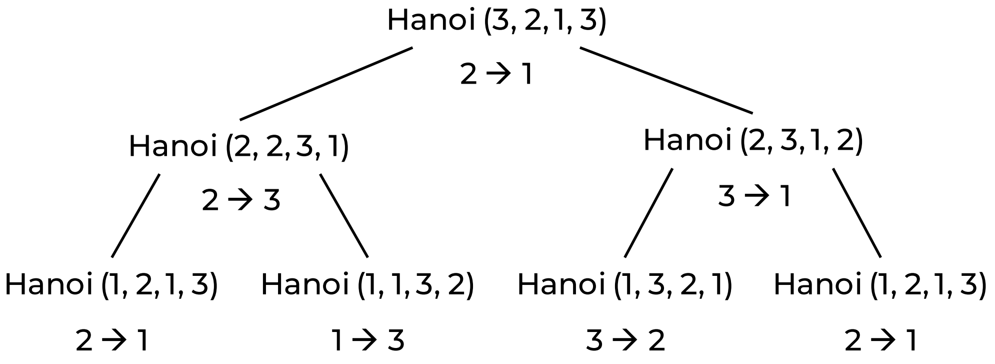
The function returns the number of disc transfers required to move the discs from the source peg to the destination peg. It also prints the sequence of disc transfers. Consider the following scenario: How would you determine the parameters `(n,s,d,a)` for the first call of the function `Hanoi`? Draw the corresponding recursive tree and list out the sequence of disc transfers from one peg to another.



Solution:

First Call: Hanoi (3, 2, 1, 3)

Recursive Tree:



Disc Transfer Sequence:

2 → 1
2 → 3
1 → 3
2 → 1
3 → 2
3 → 1
2 → 1

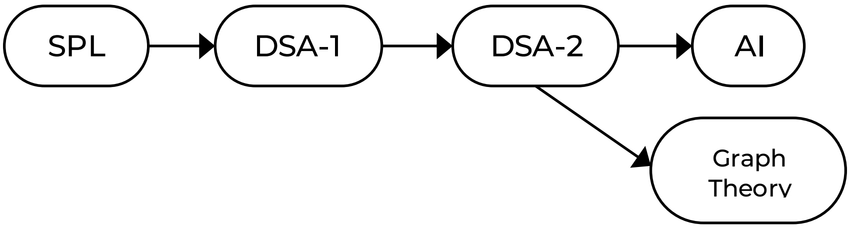
1. b) Your new trimester is beginning and you need to register for your new set of courses. While registering, you find that some courses have prerequisites. That is, some courses must be completed before others. The following is a list of prerequisites:

SPL must be completed before DSA-1.
SPL must be completed before DSA-2.
DSA-1 must be completed before DSA-2.
DSA-1 must be completed before AI.
DSA-2 must be completed before AI.
DSA-2 must be completed before Graph Theory.

Which data structure will you use to represent this set of dependencies? Draw the representation. Find a valid order that will allow you to take the courses without violating any prerequisites.

Solution:

Directed Graph will be appropriate data structure for representing prerequisites of each course.



In this graph, if we use any graph traversal algorithm, we can't visit our desired course without visit its prerequisites course(s).

1. c) Suggest a data structure for the flight landing system at Hazrat Shahjalal International Airport and justify your choice of data structure.

Scenario 1: Flight BS01, BG02, IGO4, KAL696 and CS5 are lined up for landing in the order that is given.

Scenario 2: Flight BS01, BG02, IGO4, and CS5 are lined up for landing. Meanwhile, ATC (Air Traffic Controller) heard a Mayday (dual engine failure) call from Flight KAL696 an inbound flight from Kathmandu to Seoul which is traveling over Dhaka.

Solution:

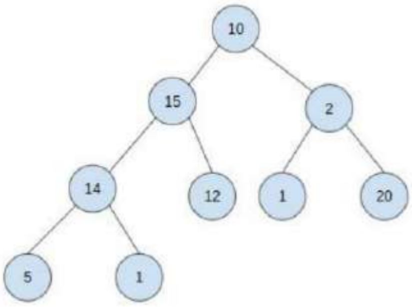
Scenario 1: Since these flights are lined up for landing in the order that is given, Queue should be implemented here. Which flight will came first, that will land first, which is similar to Queue's First-In-First-Out policy.

Scenario 2: Since some of flights have higher priority for landing first, we should implement Priority Queue here. Which flight have higher priority, that flight would be land first before other, which is similar to Priority Queue's policy.

2. a) Consider the given pseudocode of `Max-Heapify` and the following heap.

[P.T.O]

```
Max-Heapify(A,i)
    l = LEFT(i)
    r = RIGHT(i)
    if l<=A.heap-size and A[l]>A[i]
        largest=l
    else largest = i
    if r<=A.heap-size and A[r]>A[largest]
        argest = r
    if largest != i
        Exchange A[i] with A [largest]
        Max-Heapify(A,largest)
```



Does running `Max-Heapify` on the first node of the heap convert it into a **max heap**? Give a reason in favor of your answer.

If not, propose a suitable algorithm (along with the necessary pseudocode) to convert the given heap into a max heap.

Solution:

Running **`Max-Heapify`** on the first node will not convert it into max heap. If we run **`Max-Heapify`** on first node, it will exchange 10 and 15, and then it will exchange 10 and 14. It will change only left side of tree. But right side will be unchanged. In right side, parent of 20 is 2, which is breaking the condition of max heap.

But **`Build-Max-Heap`** algorithm can convert it into a max heap. **`Build-Max-Heap`** algorithm calls **`Max-Heapifiy`** for every node except leaf nodes.

Algorithm: *(Pseudcode)*

```
Build-Max-Heap(A)
    heap-size[A] ← length[A]
    for i ← [length[A]/2] downto 1
        do Max-Heapify(A, i)
```

2. b) Draw a binary tree from the following **Preorder** and **Inorder** sequences

Preorder: a b c d e f g h
Inorder: c b d a f g e h

Here, a=last digit of your student id+3, b=a+3, e=a+b, c=b+e, d=a+2, f=c+d, h=111, g=121

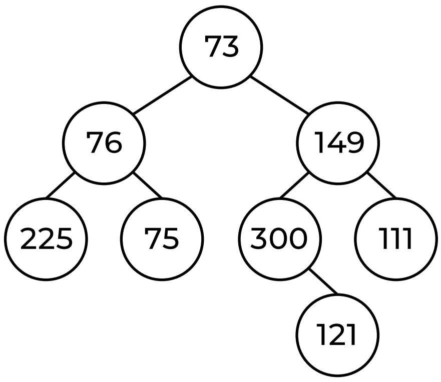
Solution:

Here, a = 70+3 = 73
b = 73+3 = 76
e = 73+76 = 149
c = 76+149 = 225

d = 73+2 = 75
f = 225+75 = 300
h = 111
g = 121

Preorder:	73	76	225	75	149	300	121	111
Inorder:	225	76	75	73	300	121	149	111

The binary tree has been drawn below:



2. c) Two disjoint sets `{y, p, z, x}` and `{r, t}` are given where minimum one of a set is the representative of that set. Determine `UNION(Find(x), Find(t))`. How can you check x and y are in the same set using Find operation? Here, x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2, t=700.

Solution:

Here, x = 70+3 = 73 p = 76+149 = 225
y = 73+3 = 76 r = 73+2 = 75
z = 73+76 = 149 t = 700

{ 76, 225, 149, 73 }
Minimum: 73

{ 75, 700 }
Minimum: 75

Representing these disjoint sets with array:

	73		75	73		73		73		75	
...	73	...	75	76	...	149	...	225	...	700	...

Now,
UNION(Find(x), Find(t))

Here,
Find(x) = Find(73) → 73
Find(t) = Find(700) → 75

After executing UNION(73, 75):

	73		73	73		73		73		73	
...	73	...	75	76	...	149	...	225	...	700	...

Again,
Find(x) = Find(73) → 73
Find(y) = Find(76) → 73

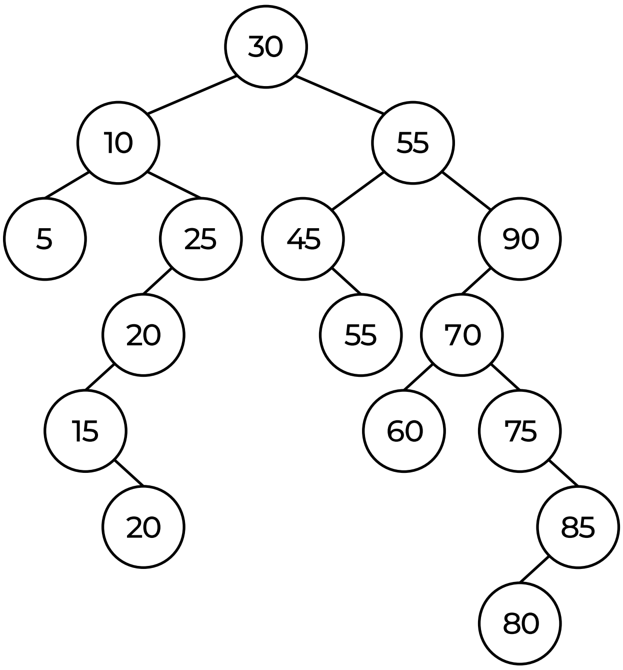
Since Find(x) and Find(y) are returning same value, therefore they are in same set.

3. a) Construct a binary search tree (**BST**) using the following nodes:

30, 55, 10, 25, 45, 5, 20, 90, 70, 60, 75, 15, 20, 85, 80, 55

Solution:

The binary search tree (BST) has been drawn below:



3. b) Traverse the constructed BST using the **inorder** traversal technique.

Solution:

The sequence of inorder traversal in constructed BST has been written below:

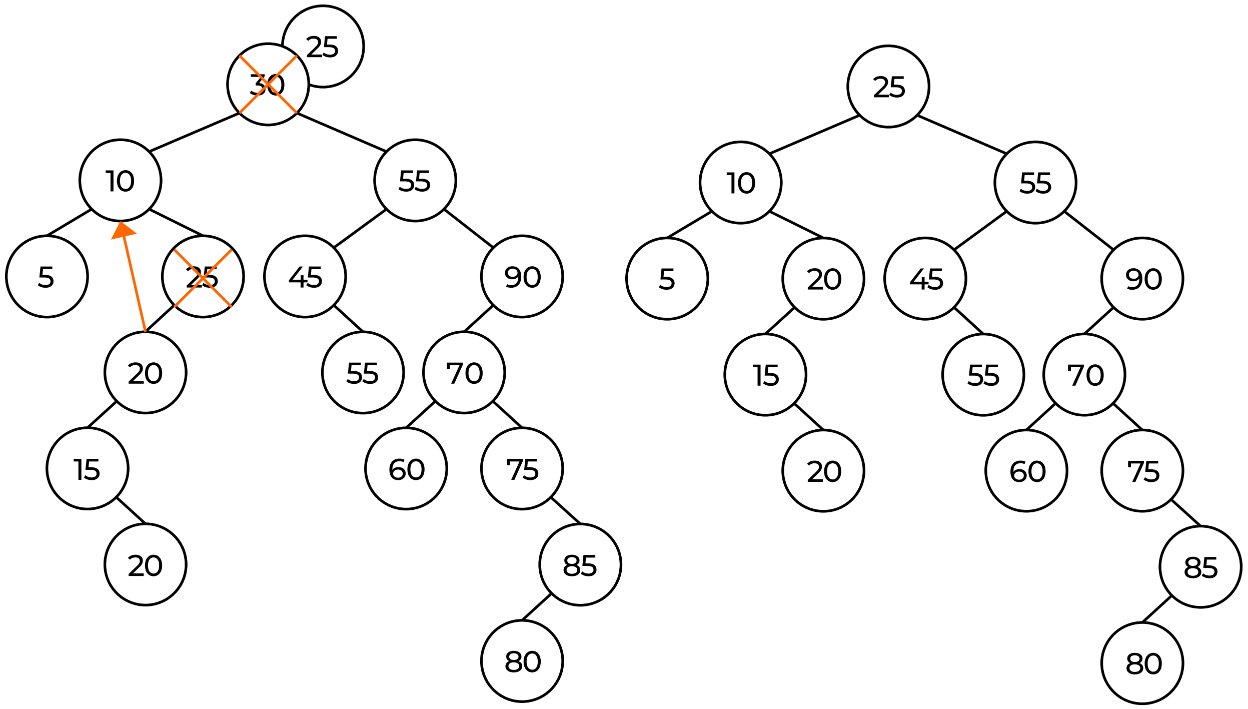
Inorder: 5, 10, 15, 20, 20, 25, 30, 45, 55, 55, 60, 70, 75, 80, 85, 90

3. c) Delete 30 and 90 respectively from the BST.

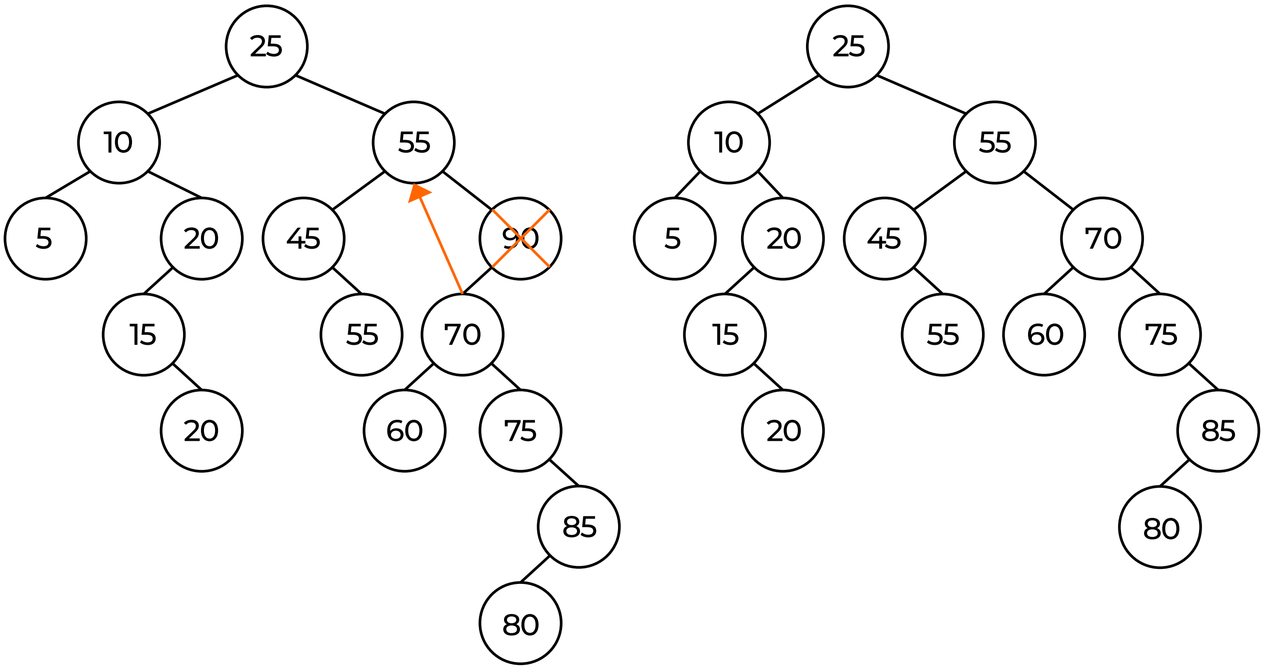
Solution:

Deleting 30:

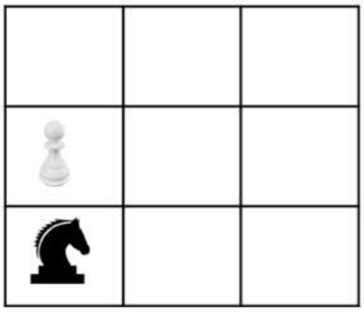
[P.T.O]



Deleting 90:



4. a) A knight moves on a chess board in an interesting way: it can jump two squares in one direction and then one square sideways. Now, draw a graph for the above chessboard in such a way that a vertex depicts each of the positions of the board. If the knight can move from one position on the board to another in a single move, draw an edge connecting the two corresponding vertices for those two positions.



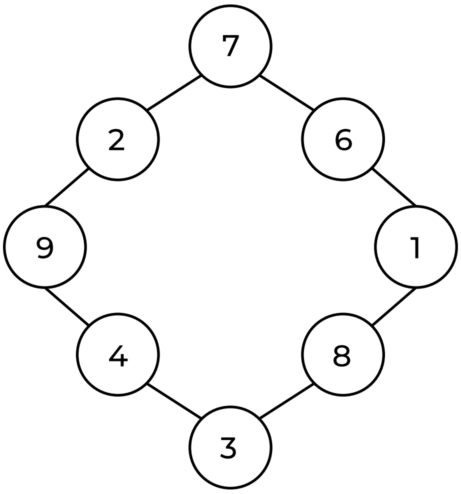
Solution:

[P.T.O]

Leveling each position of given chessboard:

1	2	3
4	5	6
7	8	9

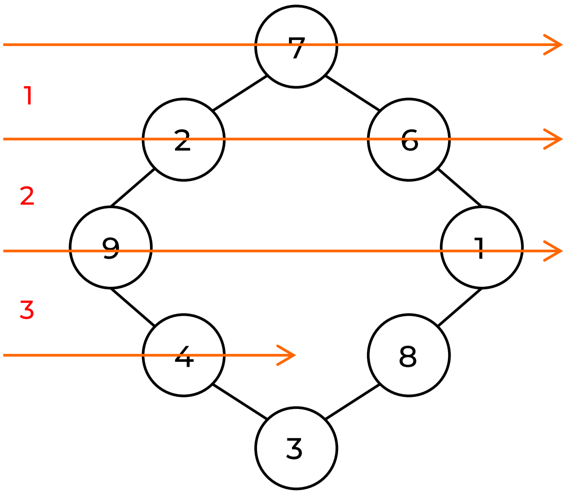
Currently knight is in position 7. The graph is drawn below, where all of position that knight can go in each move are connected by edges.



4. b) What is the minimum number of moves required by the knight to capture the pawn in the above figure? Illustrate by simulating **BFS** on the graph obtained in **Question 4(a)**.

Solution:

Current knight is in position 7 and pawn is in position 4.



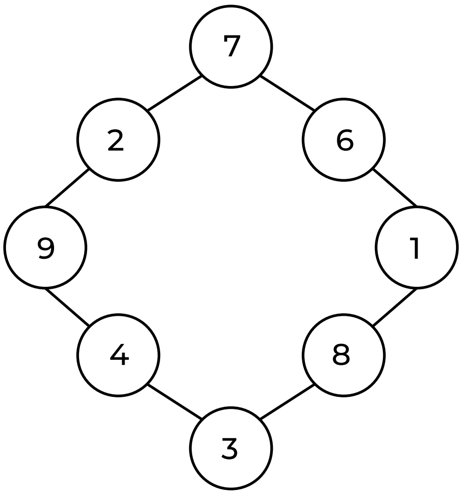
Minimum moves to capturing pawn:

7 → 2 → 9 → 4

∴ Minimum number of move required is 3.

4. c) “If the pawn were at the center position in the middle of the board, the knight would never be able to capture the pawn” - Do you agree with this statement? Justify your answer using **BFS**.

Solution:



Queue:

~~7~~	~~2~~	~~6~~	~~9~~	~~1~~	~~4~~	~~8~~	~~3~~

<u>BFS Result:</u>

7, 2, 6, 9, 1, 4, 8, 3

The center position in the middle is position 5. Applying BSF in graph from 4(a), we can see that knight will never visit position 5.

∴ If the pawn were at the center position in middle of the board, it was not possible for knight to capture the pawn.

Spring 2023

1. a) Convert the following infix expression into postfix using a STACK.

Infix expression: a↑2-(b+c/d*a+b)

Solution:

Repeat of Fall 2023 Question 1(b)

1. b) Evaluate the postfix expression, a b – c d * + for a=2, b=3, c=2 and d=1 using a STACK

Solution:

Repeat of Fall 2023 Question 1(c)

1. c) Find a recursive algorithm for TOWER OF HANOI using one intermediate pillar/peg and show simulation for n = 3, where n is the number of disks.

Solution:

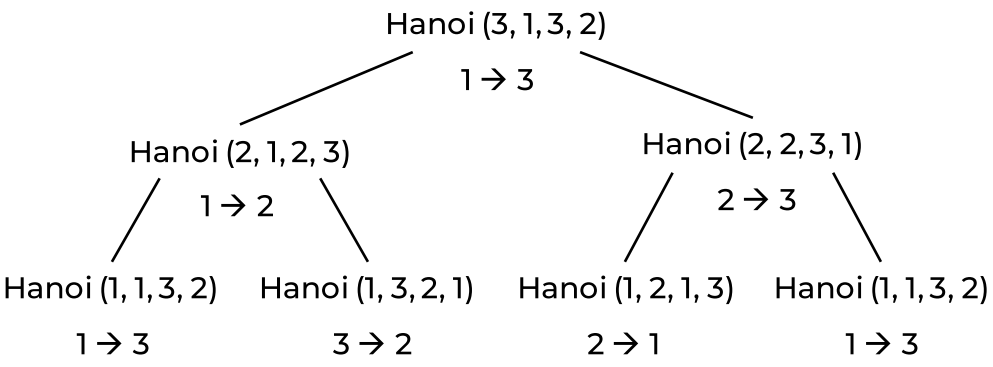
Algorithm:

```
void towerOfHanoi(int n, int s, int d, int a) {
    if(n==1) {
        printf("%d -> %d", s, d);
        return;
    }

    towerOfHanoi(n-1,s,a,d);
    printf("%d -> %d", s, d);
    towerOfHanoi(n-1,a,d,s);
}
```

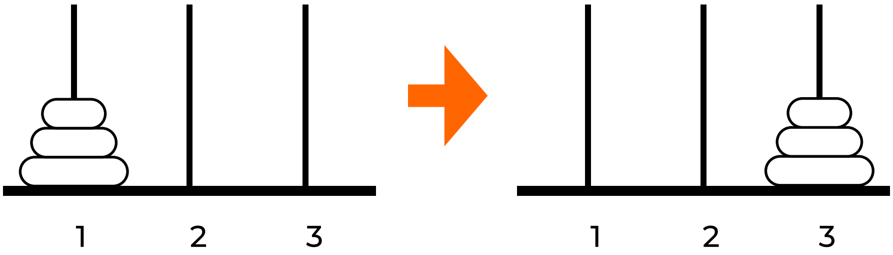
Simulation:

Let,
Staring peg, s = 1
Destination peg, d = 3
Auxiliary peg, a = 2



Disc Transfer Sequence:

1 → 3
1 → 2
3 → 2
1 → 3
2 → 1
2 → 3
1 → 3



1. d) How overflow checking in a QUEUE is done when it is implemented by an array?

Solution:

We can implement Queue in array in two ways, normally and circularly.

For Normal Queue, we can check overflow by this condition:

```
if (rare == size-1) {
    //Overflow
}
```

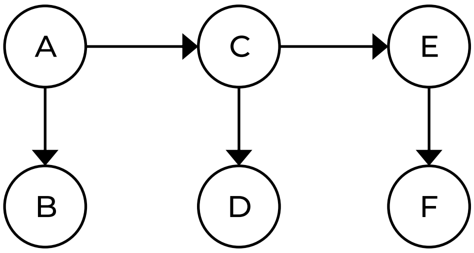
For Circular Queue, we can check overflow by this condition:

```
if ((rare == size-1 && front == 0) || (front == (rare+1)%size)) {
    //Overflow
}
```

2. a) Draw a directed acyclic graph using six vertices.

Solution:

The acyclic graph has been drawn below:



2. b) Construct an Adjacency Matrix and an Adjacency List for the graph in Ques. 2(a). Show the memory requirement for each of the cases.

Solution:

[P. T. O]

Adjacency Matrix:

$$\begin{array}{c|cccccc|} & A & B & C & D & E & F \\ \hline A & 0 & 1 & 1 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 1 & 1 & 0 \\ D & 0 & 0 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 1 \\ F & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Memory Requirement = $O(V^2)$

Here, V = Number of Vertices

Adjacency List:

A → B, C
B →
C → D, E
D →
E → F
F →

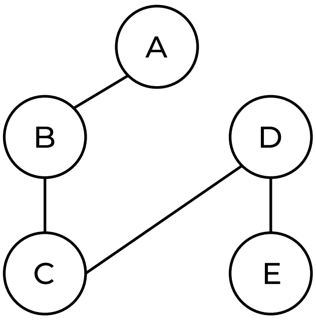
Memory Requirement = $O(|V|+|E|)$

Here, V = Number of Vertices
E = Number of Edges

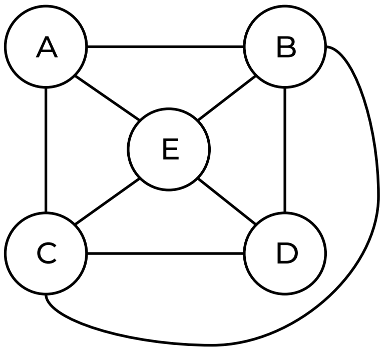
2. c) Sketch a sparse and a dense graph using five vertices.

Solution:

Sparse Graph:



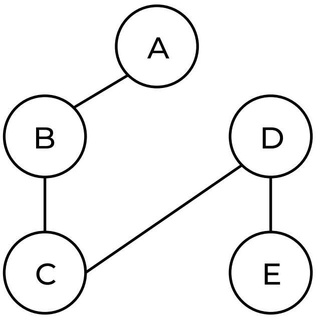
Dense Graph:

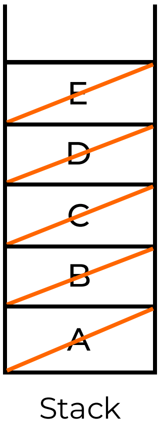


2. d) Show the depth first search (DFS) sequence from the sparse graph of 2(c) assuming any one is the starting vertex.

Solution:

Let, starting element is A





DFS Result:

A, B, C, D, E

∴ Final DFS Sequence is:
A, B, C, D, E

2. e) Write an algorithm for Topological Sorting. Show the simulation of your algorithm using the graph in Ques. 2(a).

Solution:

Algorithm: *(Pseudocode)*

```
TopologicalSort(Graph G)
    Queue Q
    LinkedList L

    for G.vertex.length() downto 1
        if G.vertex.inDegree == 0
            Q.enqueue(G.vertex)

    while Q.isEmpty() is false
        Vertex V = Q.dequeue()
        L.add(V)
        foreach G.neighborsOf(V) as U
            U.inDegree = U.inDegree-1
            if U.inDegree == 0
                Q.enqueue(U)

    return L
```

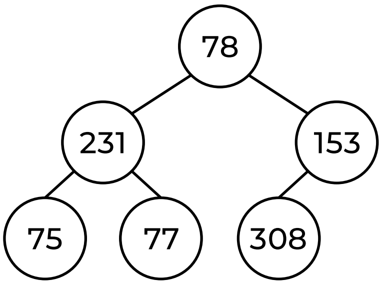
Simulation:

Repeat of ***Fall 2023 Question 3(b)***

3. a) Draw a binary tree that contains six nodes.

Solution:

The binary tree has been drawn below:



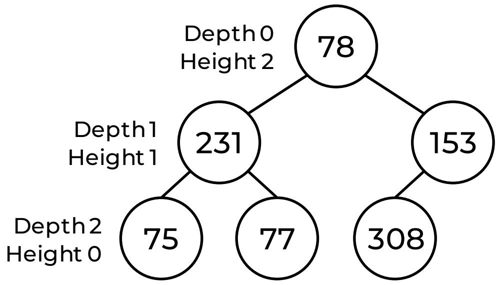
3. b) Traverse the binary tree of Ques. 3(a) using the preorder, inorder, postorder and level order techniques. Level each of the nodes of the tree. Also find the height of the tree using level.

Solution:

Preorder:	78, 231, 75, 77, 153, 308
Inorder:	75, 231, 77, 78, 308, 153
Postorder:	75, 77, 231, 308, 153, 78
Level Order:	78, 231, 153, 75, 77, 308

Now, leveling the tree:

[P.T.O]

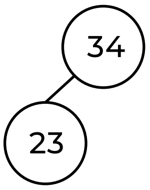
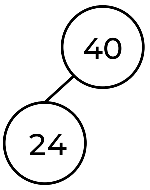
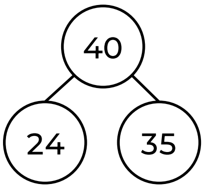
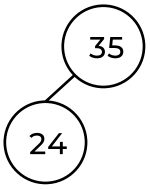


∴ Height of the tree is 2.

3. c) Show the status of a QUEUE and a Priority QUEUE (Data in descending Order) for the following operations, where both QUEUEs are implemented by an array of size, m=3. Here, Enqueue and Dequeue mean insert and delete respectively

Enqueue(23), Enqueue(34), Dequeue(), Enqueue(40), Enqueue(35), Dequeue()

Solution:

	Queue	**Priority Queue**
Enqueue(23)	23	
Enqueue(34)	23 34	
Dequeue()	34	
Enqueue(40)	34 40	
Enqueue(35)	34 40 35	
Dequeue()	40 35	

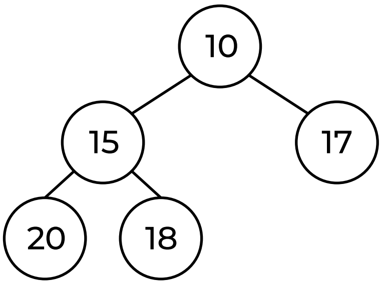
3. c) Sort the following data in descending order using the heapsort algorithm.

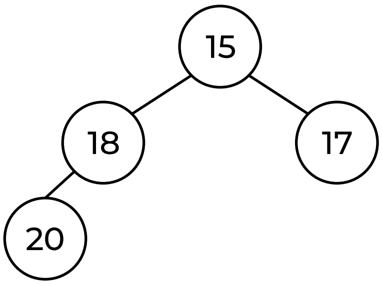
10 18 17 20 15

Solution:

Constructing min-heap:

10	15	17	20	18

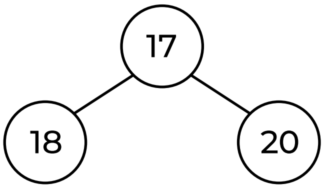




Deleting 10:

15	18	17	20	10

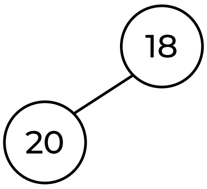
Unsorted | Sorted



Deleting 15:

17	18	20	15	10

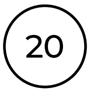
Unsorted | Sorted



Deleting 17:

18	20	17	15	10

Unsorted | Sorted



Deleting 18:

20	18	17	15	10

Unsorted | Sorted

Deleting 20:

20	18	17	15	10

Sorted

∴ Final Sorted Data is 20, 18, 17, 15, 10.

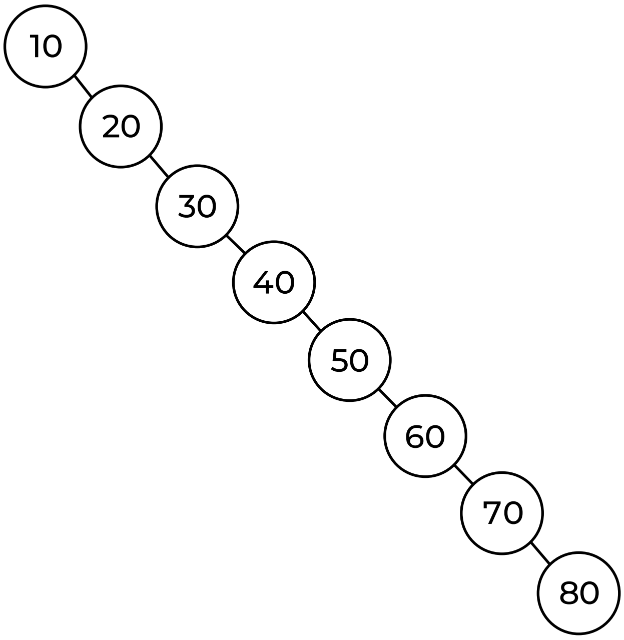
4. a) Construct a binary search tree (BST) using the nodes

10 20 30 40 50 60 70 80

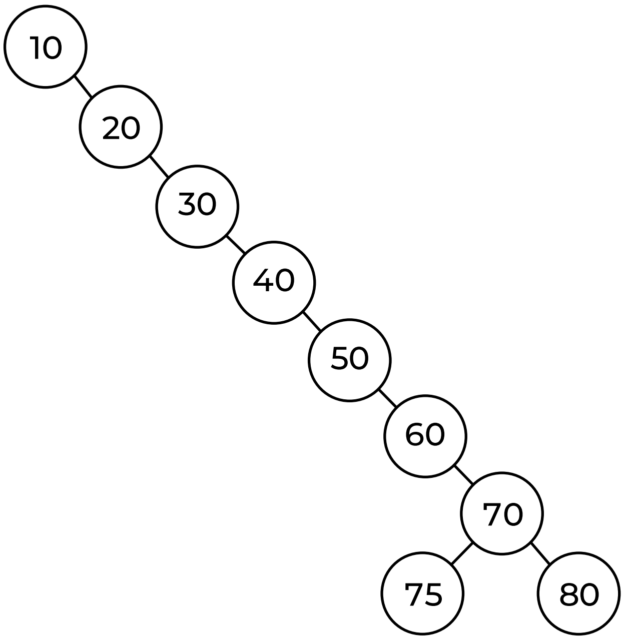
Solution:

The binary search tree (BST) has been drawn below:

[P.T.O]

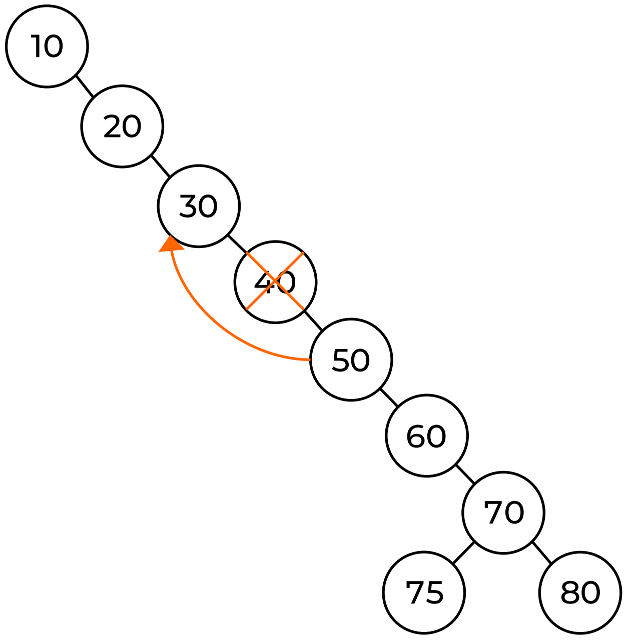


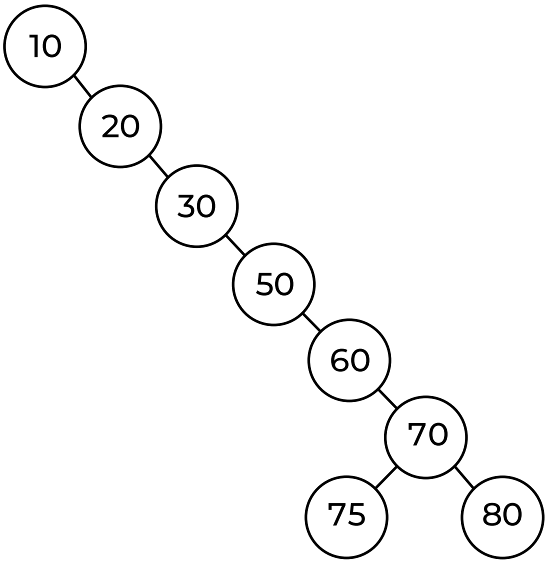
Inserting 75:



Deleting 40:

[P.T.O]

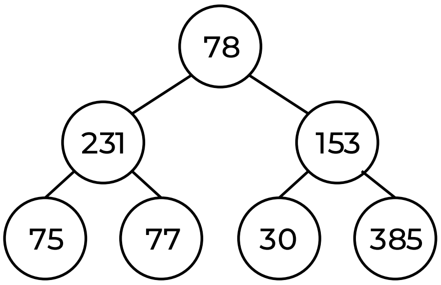




4. b) Represent a binary tree using a one-dimensional array and a linked list.

Solution:

Let, out binary tree is:

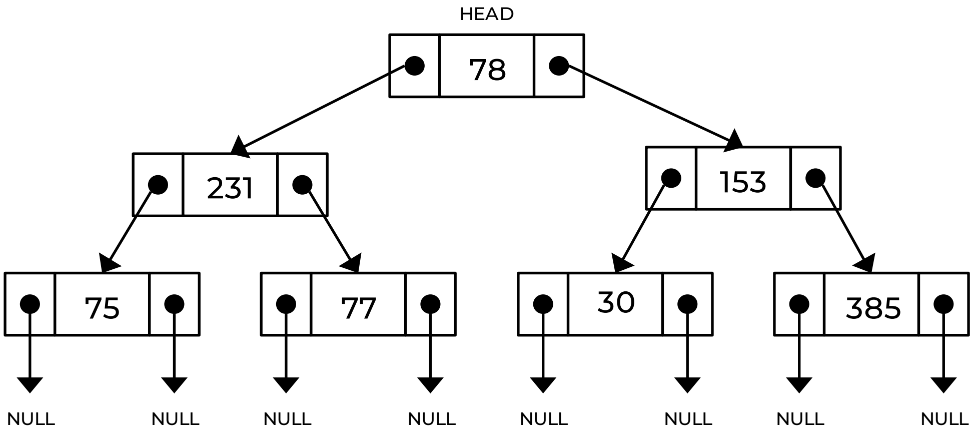


One Dimensional Array Representation:

78	231	153	75	77	308	385
0	1	2	3	4	5	6

Here,
Left Child = i*2+1
Right Child = i*2+2
Parent = ceil(i/2)-1

Linked List Representation:



4. c) S1 = {10, 20, 30, 40} and S2= {50, 60, 70} are disjoint sets, where the maximum of both sets are the representatives. How can you check 10 and 50 are not in the same set?

Solution:

{ 10, 20, 30, 40 }
Maximum: 40

{ 50, 60, 70 }
Maximum: 70

Representing these disjoint sets with array with their maximum representatives:

	40		40		40		40		70		70		70	
...	10	...	20	...	30	...	40	...	50	...	60	...	70	....

Here,

Find(10) → 40
Find(50) → 70

Since, Find(10) and Find(50) are returning different value, therefore 10 and 50 are not in the same set.

4. d) Which Data Structures are appropriate to implement the following and why?

i) Different areas of Dhaka City with distance
ii) Line in front of a Doctor's Chamber
iii) Reversing a String

Solution:

i) Different areas of Dhaka City with a distance can be representing with Weighted Graph. The vertices will represent the areas name and edges weight will be represent the distance of both area.
ii) Line in front of a Doctor's chamber can be represented by Queue because who came in the doctor's chambers line first, he/she get the service first which follow Queue's First-In-First-Out policy
iii) Stack is appropriate Data Structure to implement String Reversion. We just need to push all the character of String in a Stack and pop up and store these character one by on to reversing the String.

Fall 2022

1. a) Draw a binary tree using the data given below, where x, y, z, p, r, t, u and v are nodes of the tree.

y p z x r t u v

Here, x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r, u=600, v=700

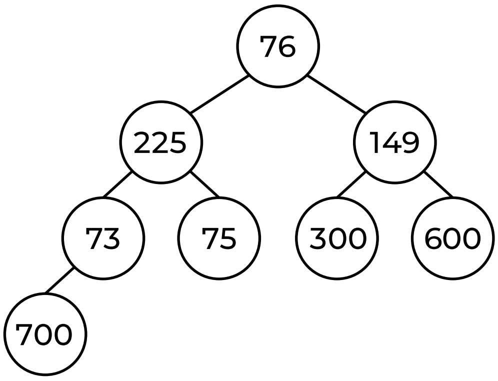
Solution:

Here, x = 70+3 = 73
y = 73+3 = 76
z = 73+76 = 149
p = 76+149 = 225

r = 73+2 = 75
t = 225+75 = 300
u = 600
v = 700

76	225	149	73	75	300	600	700
y	p	z	x	r	t	u	v

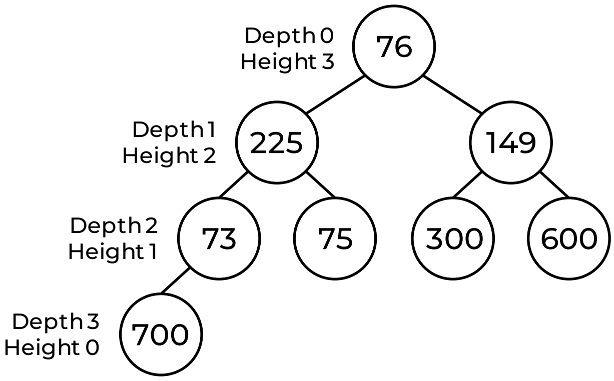
Now, the binary tree has been drawn below:



1. b) Traverse the binary tree of Ques. 1(a) using the inorder, postorder and level order techniques. Level each of the nodes of the tree. Also find the height of the tree using level.

Solution:

Leveling the tree:



∴ The height of the tree is 3.

Now,

Inorder:	700, 73, 225, 75, 76, 300, 149, 600
Postorder:	700, 73, 75, 225, 300, 600, 149, 76
Level Order:	76, 225, 149, 73, 75, 300, 600, 700

1. c) Draw a binary tree from the following Preorder and Inorder sequences

Preorder: x y p r z t v u
Inorder: p y r x t v z u

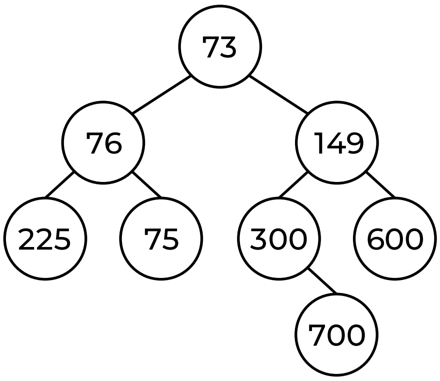
Here, x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r, u=600, v=700

Solution:

Here,	x = 70+3 = 73	r = 73+2 = 75
	y = 73+3 = 76	t = 225+75 = 300
	z = 73+76 = 149	u = 600
	p = 76+149 = 225	v = 700

Preorder:	73	76	225	75	149	300	700	600
Inorder:	225	76	75	73	300	700	149	600

The binary tree has been drawn below:



1. d) Write an algorithm for preorder traversal. Show the simulation for the tree in Ques. 1(a).

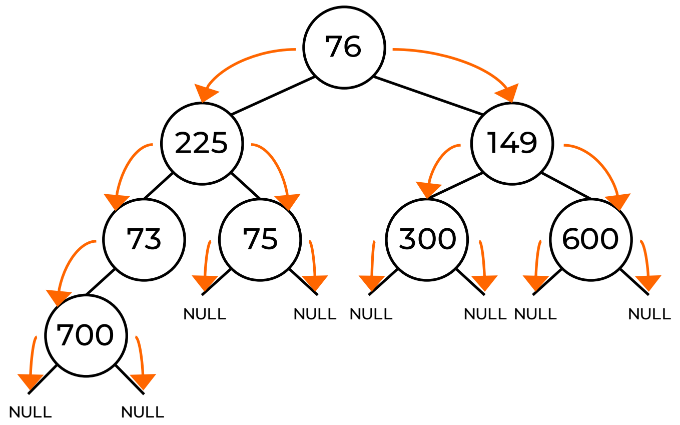
Solution:

Algorithm:

```
void preorder(struct node* root) {
    if(root == NULL) {
        return;
    }

    printf("%d", root -> value));
    preorder(root -> left);
    preorder(root -> right);
}
```

Simulation:



Preorder:

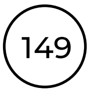
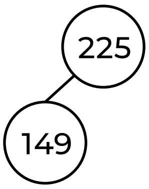
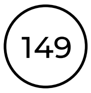
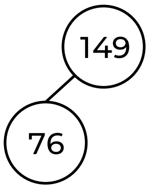
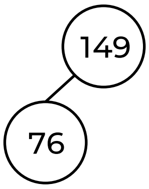
76, 225, 73, 700, 75, 149, 300, 600

2. a) Show the status of a QUEUE and a Priority QUEUE (Data in Descending Order) for the following operations, where both QUEUEs are implemented by an array of size m=3. Here, Enqueue and Dequeue mean insert and delete respectively, and x= last two digits of your student id+3, y=x+3, z=x+y and p=y+z.

Enqueue(z), Enqueue(p), Dequeue(), Enqueue(y), Enqueue(z), Dequeue()

Solution:

Here, x = 70+3 = 73 z = 73+76 = 149
y = 73+3 = 76 p = 76+149 = 225

	Queue	**Priority Queue**
Enqueue(z)	149	
Enqueue(p)	149 225	
Dequeue()	225	
Enqueue(y)	225 76	
Enqueue(z)	225 76 149	 [Duplicate elements normally can't be inserted in Priority Queue]

Dequeue()	76 149	76

2. b) Draw a complete binary tree and then build the max-heap tree from the following data, where x= last digit of your student id+150, y=x+130, z=x+y.
Finally, sort the data in ascending order using the heapsort algorithm.

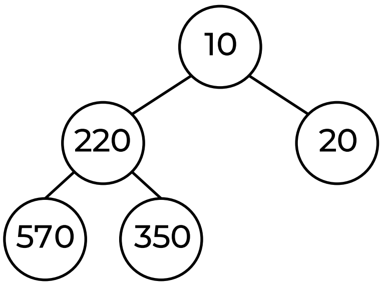
10 x 20 z y

Solution:

Here, x = 70+150 = 220
y = 220+130 = 350
z = 220+350 = 570

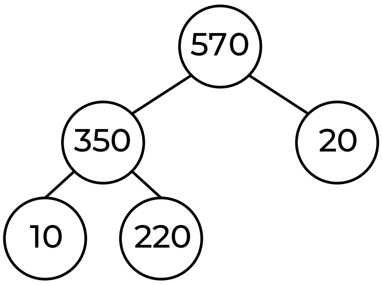
Constructing binary tree

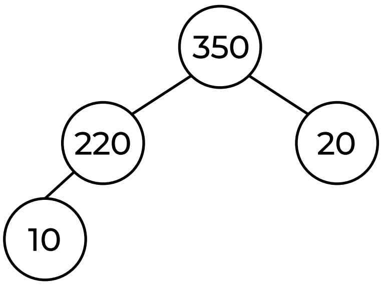
10	220	20	570	350



Constructing max-heap:

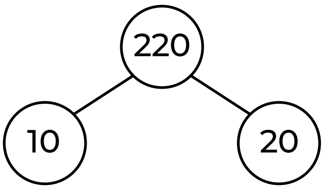
570	350	20	10	220





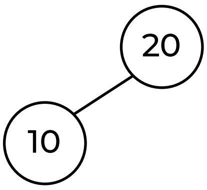
Deleting 570:

350	220	20	10	570
Unsorted				Sorted



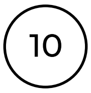
Deleting 350:

220	10	20	350	570
Unsorted			Sorted	



Deleting 220:

10	20	220	350	570
Unsorted		Sorted		



Deleting 20:

10	20	220	350	570

Unsorted | Sorted

Deleting 10:

10	20	220	350	570

Sorted

∴ Final Sorted Data is 10, 20, 220, 350, 570.

2. c) Two disjoint sets {y, p, z, x} and {r, t} are given where minimum one of a set is the representative of that set. Determine UNION(Find(x), Find(t)). How can you check x and y are in the same set using Find operation? Here, x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2, t=700.

Solution:

Repeat of ***Summer 2023 Question 2(c)***

3. a) Draw a directed acyclic graph using the vertices y, p, z, x, r and u, where x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2, u=p+r

Solution:

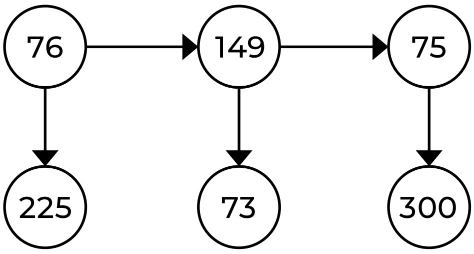
Here, $x = 70+3 = 73$ $\quad$ $p = 76+149 = 225$

$y = 73+3 = 76$ $\quad$ $r = 73+2 = 75$

$z = 73+76 = 149$ $\quad$ $u = 225+75 = 300$

76	225	149	73	75	300
y	p	z	x	r	u

Now, the acyclic graph has been drawn below:



3. b) Construct an Adjacency Matrix and an Adjacency List for the graph in Ques. 3(a).

Solution:

[P. T. O]

Adjacency Matrix:

$$
\begin{array}{c|cccccc|}
 & 76 & 225 & 149 & 73 & 75 & 300 \\
\hline
76 & 0 & 1 & 1 & 0 & 0 & 0 \\
225 & 0 & 0 & 0 & 0 & 0 & 0 \\
149 & 0 & 0 & 0 & 1 & 1 & 0 \\
73 & 0 & 0 & 0 & 0 & 0 & 0 \\
75 & 0 & 0 & 0 & 0 & 0 & 1 \\
300 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$

Adjacency List:

76	→	225, 149
225	→	
149	→	73, 75
73	→	
75	→	300
300	→	

3. c) Write an algorithm for Topological Sorting. Show the simulation of your algorithm using the graph in Ques. 3(a).

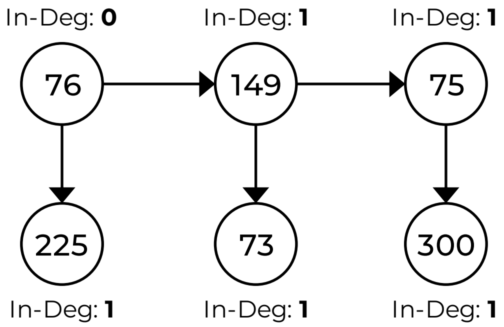
Solution:

Algorithm:

*Repeat of **Spring 2023 Question 2(e)***

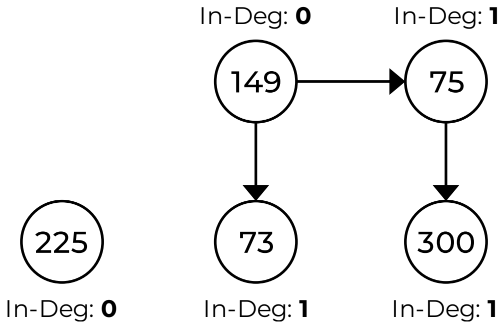
Simulation:

Simulation of Topological Ordering Algorithm has been showed below:



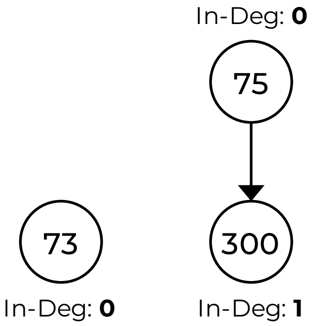
Topological Order:

76



Topological Order:

76, 225, 149



Topological Order:

76, 225, 149, 73, 75

<u>Topological Order:</u>

76, 225, 149, 73, 75, 300

∴ Final Topological Order is 76, 225, 149, 73, 75, 300.

3. **d)** Sketch a sparse and a dense graph using vertices y, p, z, x and r where x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2

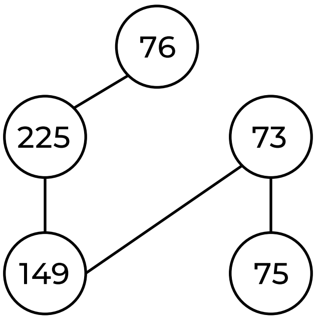
Solution:

Here, x = 70+3 = 73 p = 76+149 = 225
y = 73+3 = 76 r = 73+2 = 75
z = 73+76 = 149

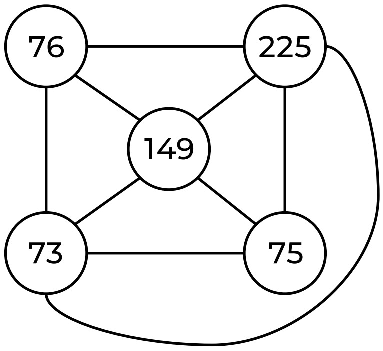
76	225	149	73	75
y	p	z	x	r

Now, the Sparse Graph and Dense Graph have been drawn below:

Sparse Graph:



Dense Graph:



4. **a)** Draw an undirected graph using the vertices y, p, z, x and r, where x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2. Also find the Breadth First Search (BFS) sequence from the graph considering x is starting vertex

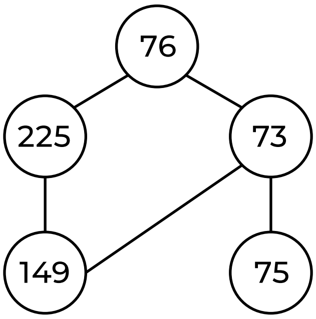
Solution:

Here, x = 70+3 = 73 p = 76+149 = 225
y = 73+3 = 76 r = 73+2 = 75
z = 73+76 = 149

76	225	149	73	75
y	p	z	x	r

Now, the undirected graph has been drawn below:

[P. T. O]



Considering x (73) as starting vertex:

Queue:

~~73~~	~~75~~	~~76~~	~~149~~	~~225~~

BFS Result:

73, 75, 76, 149, 225

∴ BFS sequence is 73, 75, 76, 149, 225.

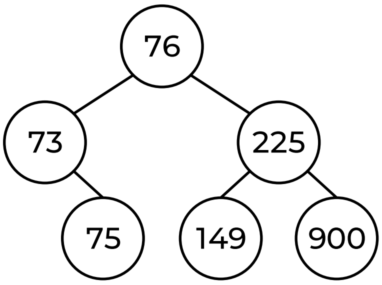
4. b) Construct a binary search tree (BST) using the nodes y, p, z, x, r and t, where x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2, t=900. Show the insertion and deletion of p+r and z, respectively in/from the BST.

Solution:

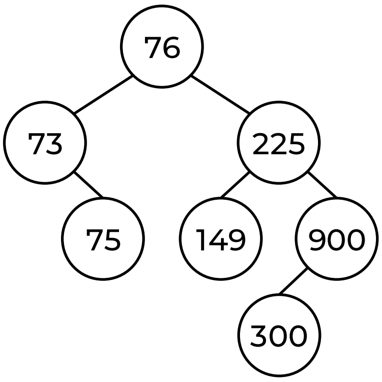
Here,	x = 70+3 = 73	p = 76+149 = 225
	y = 73+3 = 76	r = 73+2 = 75
	z = 73+76 = 149	t = 900

76	225	149	73	75	900
y	p	z	x	r	t

The binary search tree (BST) has been drawn below:

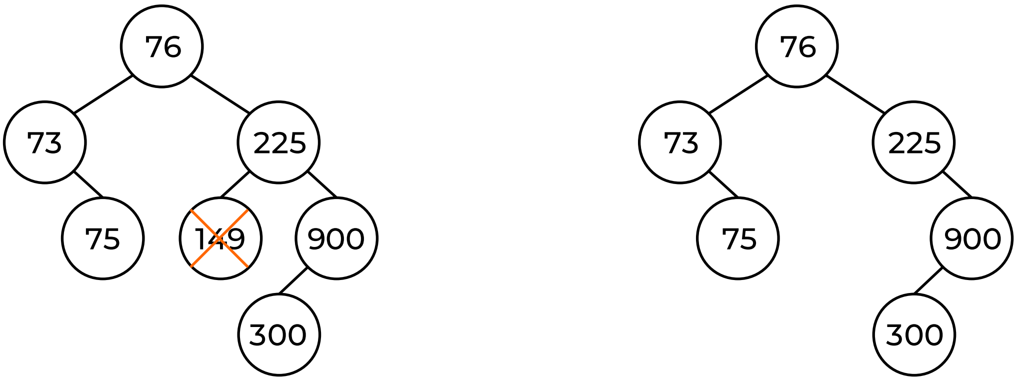


Inserting p+r: *(225+75 = 300)*



Deleting z: *(149)*

[P.T.O]



4. c) Show the space requirement to represent a directed graph using a 2D array and a linked list.

Solution:

To representing an undirected graph with 2D Array, which has V vertices with any number of edges, we need to make a $V \times V = V^2$ size array.
∴ Memory requirement for this case is **O(V^2).**

To representing an undirected graph with Linked List, which has V vertices and E edges, we make V size array which contain V Linked List and these all Linked List have total E elements within them.
∴ Memory requirement for this case is **O(|V|+|E|).**

4. d) Which Data Structures are appropriate to implement the following and why?

i) Different areas of Dhaka City with distance
ii) Infix to Postfix Conversion
iii) Breadth First Search

Solution:

i) Different areas of Dhaka City with a distance can be representing with Weighted Graph. The vertices will represent the areas name and edges weight will be represent the distance of both area.
ii) We can implement Infix to Postfix conversion using stack. While conversion, we should print all symbol and if we get any operator, we should push it in stack and pop and print it when our condition match.
iii) Queue is the appropriate data structure for implement Breadth First Search (BFS). In BFS, we start our functionality by enqueuing starting the vertex and then we dequeue and print the vertices and enqueue their neighbor vertices one by one.

Summer 2022

1. a) Draw a binary tree using the data given below, where x, y, z, p, r, t, u and v are nodes of the tree.

y p z x r t u v

Here, x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r, u=600, v=700

Solution:

Here, x = 70+5 = 75

y = 75+3 = 78

z = 75+78 = 153

p = 78+153 = 231

r = 75+2 = 77

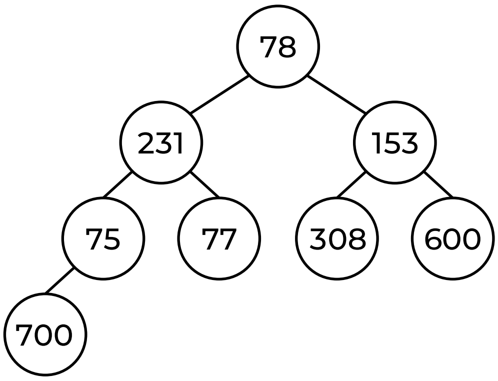
t = 231+77 = 308

u = 600

v = 700

78	231	153	75	77	308	600	700
y	p	z	x	r	t	u	v

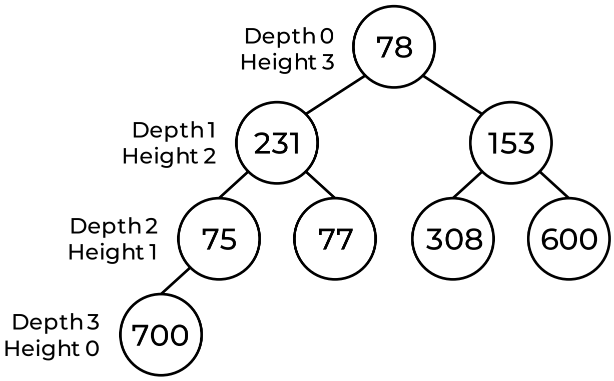
Now, the binary tree has been drawn below:



1. b) Traverse the binary tree of Ques. 1(a) using the preorder, inorder and postorder techniques. Level each of the nodes of the tree. Also find the height of the tree using level.

Solution:

Leveling the tree:



∴ The height of the tree is 3.

Now,

Preorder: 78, 231, 75, 700, 77, 153, 308, 600

Inorder: 700, 75, 231, 77, 78, 308, 153, 600

Postorder: 700, 75, 77, 231, 308, 600, 153, 78

1. c) Draw a binary tree from the following Preorder and Inorder sequences

Preorder: x y p r z t v u
Inorder: p y r x t v z u

Here, x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, t=p+r, u=600, v=700

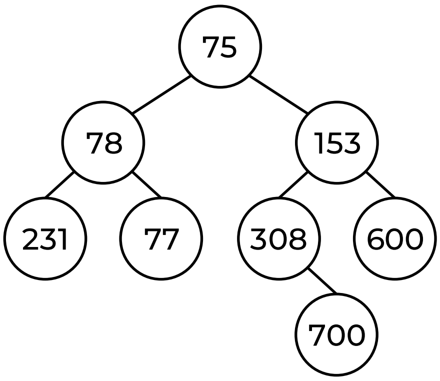
Solution:

Here, x = 70+5 = 75
y = 75+3 = 78
z = 75+78 = 153
p = 78+153 = 231

r = 75+2 = 77
t = 231+77 = 308
u = 600
v = 700

Preorder:	75	78	231	77	153	308	700	600
Inorder:	231	78	77	75	308	700	153	600

The binary tree has been drawn below:



1. d) Write an algorithm for level order. Show the simulation for the tree in Ques. 1(a).

Solution:

Algorithm:

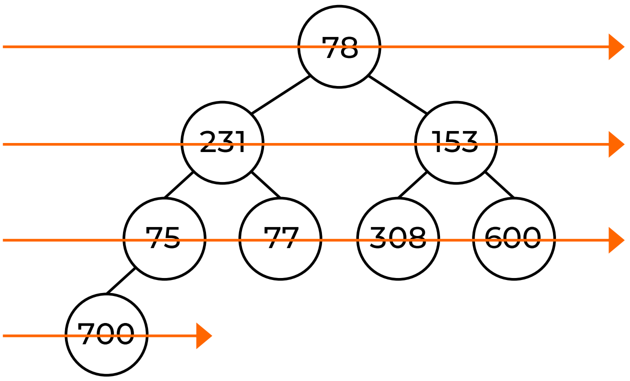
```
void levelorder(struct node* root) {
    Queue queue = new Queue();
    queue.enqueue(root);

    while(queue.isEmpty() == false) {
        currentNode = queue.dequeue();
        printf("%d ", currentNode -> value);

        if (currentNode->left != NULL) {
            queue.enqueue(currentNode -> left);
        }
```

```
            if (currentNode->right != NULL) {
                queue.enqueue(currentNode -> right);
            }
        }
    }
```

Simulation:



Level Order:

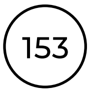
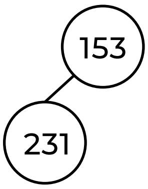
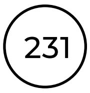
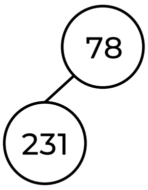
78, 231, 153, 75, 77, 308, 600, 700

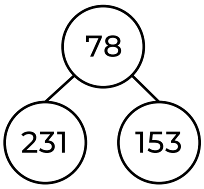
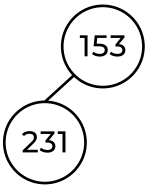
2. a) Show the status of a QUEUE and a Priority QUEUE (Data in Ascending Order) for the following operations, where both QUEUEs are implemented by an array of size m=3. Here, Enqueue and Dequeue mean insert and delete respectively, and x= last two digits of your student id+5, y=x+3, z=x+y and p=y+z.

Enqueue(z), Enqueue(p), Dequeue(), Enqueue(y), Enqueue(z), Dequeue()

Solution:

Here, x = 70+5 = 75 z = 75+78 = 153
y = 75+3 = 78 p = 78+153 = 231

	Queue	Priority Queue
Enqueue(z)	153	
Enqueue(p)	153 231	
Dequeue()	231	
Enqueue(y)	231 78	

Enqueue(z)	231 | 78 | 153	
Dequeue()	78 | 153	

2. **b)** Draw a complete binary tree and then build the min-heap tree from the following data, where x= last digit of your student id+150, y=x+130, z=x+y.
Finally, sort the data in descending order using the heapsort algorithm.

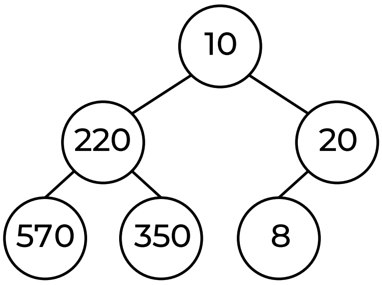
10 x 20 z y 8

Solution:

Here, x = 70+150 = 220 y = 220+130 = 350
z = 220+350 = 570

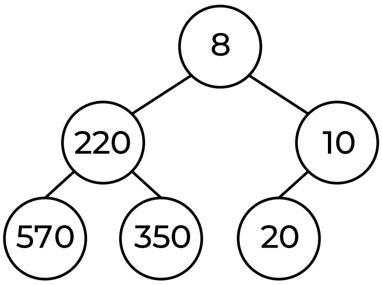
Constructing binary tree

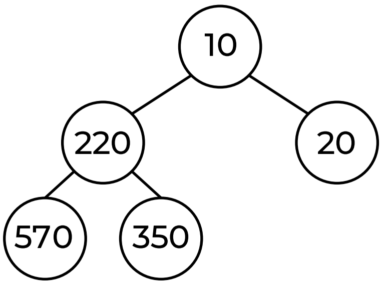
10	220	20	570	350	8



Constructing min-heap:

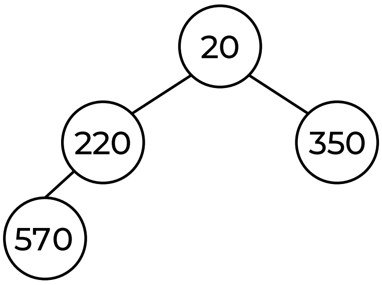
8	220	10	570	350	20





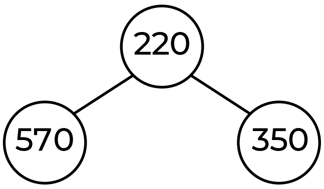
Deleting 8:

10	220	20	570	350	8
		Unsorted			Sorted



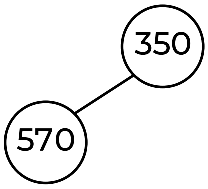
Deleting 10:

20	220	350	570	10	8
	Unsorted				Sorted



Deleting 20:

20	220	350	20	10	8
Unsorted			Sorted		



Deleting 220:

570	350	220	20	10	8
Unsorted		Sorted			

Deleting 350:

570	350	220	20	10	8
Unsorted	Sorted				

Deleting 570:

570	350	220	20	10	8
Sorted					

∴ Final Sorted Data is 570, 350, 220, 20, 10, 8.

2. c) Two disjoint sets {y, p, z, x} and {r, t} are given where maximum one of a set is the representative of that set. Determine UNION(Find(x), Find(t)). How can you check x and y are in the same set using Find operation? Here, x=last digit of your student id+9, y=x+3, z=x+y, p=y+z, r=x+2, t=700.

Solution:

Here,	x = 70+9 = 79	p = 82+161 = 243
	y = 79+3 = 82	r = 79+2 = 81
	z = 79+82 = 161	t = 700

{ 82, 243, 161, 79 }
Maximum: 243

{ 81, 700 }
Maximum: 700

Representing these disjoint sets with array:

	243		700	243		243		243		700	
...	79	...	81	82	...	161	...	243	...	700	...

Now,

UNION(Find(x), Find(t))

Here,

Find(x) = Find(79) → 243
Find(t) = Find(700) → 700

After executing UNION(75, 77):

	700		700	700		700		700		700	
...	79	...	81	82	...	161	...	243	...	700	...

Again,

Find(x) = Find(79) → 700
Find(y) = Find(82) → 700

Since Find(x) and Find(y) are returning same value, therefore they are in same set.

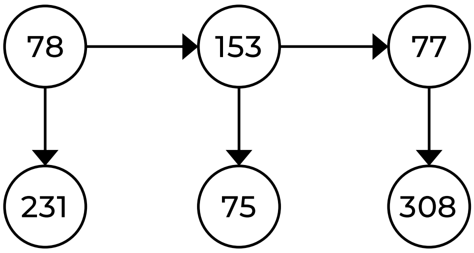
3. a) Draw a directed acyclic graph using the vertices y, p, z, x, r and u, where x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, u=p+r

Solution:

Here, x = 70+5 = 75
y = 75+3 = 78
z = 75+78 = 153
p = 78+153 = 231
r = 75+2 = 77
u = 231+77 = 308

78	231	153	75	77	308
y	p	z	x	r	u

Now, the acyclic graph has been drawn below:



3. b) Construct an Adjacency Matrix and an Adjacency List for the graph in Ques. 3(a).

Solution:

Adjacency Matrix:

$$\begin{array}{c|cccccc} & 78 & 231 & 153 & 75 & 77 & 308 \\ \hline 78 & 0 & 1 & 1 & 0 & 0 & 0 \\ 231 & 0 & 0 & 0 & 0 & 0 & 0 \\ 153 & 0 & 0 & 0 & 1 & 1 & 0 \\ 75 & 0 & 0 & 0 & 0 & 0 & 0 \\ 77 & 0 & 0 & 0 & 0 & 0 & 1 \\ 308 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Adjacency List:

76	→	231, 153
231	→	
153	→	75, 77
76	→	
77	→	300
308	→	

3. c) Write an algorithm for Topological Sorting. Show the simulation of your algorithm using the graph in Ques. 3(a).

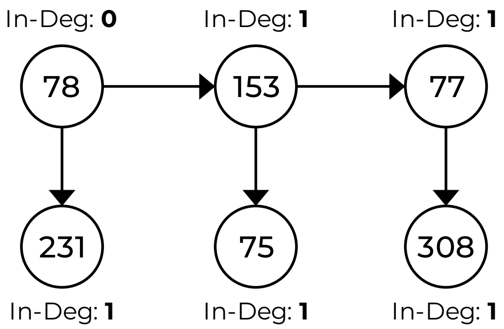
Solution:

Algorithm:

*Repeat of **Spring 2023 Question 2(e)***

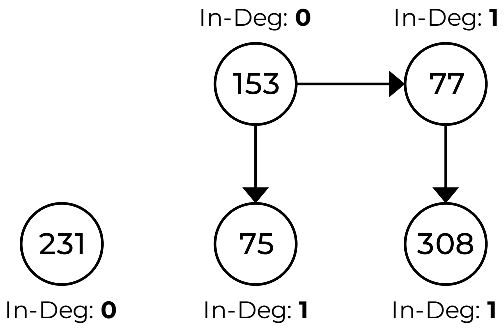
Simulation:

Simulation of Topological Ordering Algorithm has been showed below:



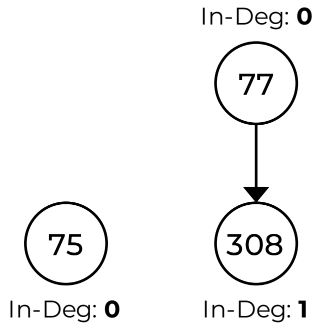
<u>Topological Order:</u>

78



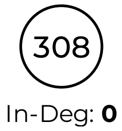
<u>Topological Order:</u>

78, 231, 153



<u>Topological Order:</u>

78, 231, 153, 75, 77



<u>Topological Order:</u>

78, 231, 153, 75, 77, 308

∴ Final Topological Order is 78, 231, 153, 75, 77, 308.

3. d) Sketch a sparse and a dense graph using vertices y, p, z, x and r where x=last digit of your student id+3, y=x+3, z=x+y, p=y+z, r=x+2

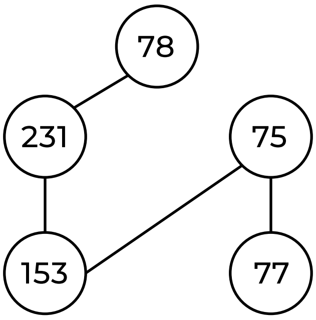
Solution:

Here, x = 70+5 = 75
y = 75+3 = 78
z = 75+78 = 153
p = 78+153 = 231
r = 75+2 = 77

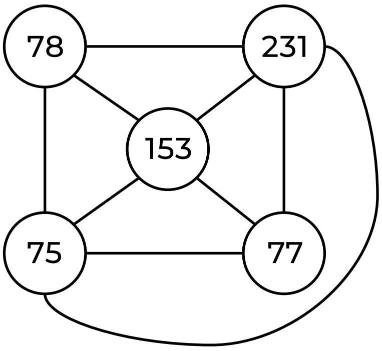
78	231	153	75	77
y	p	z	x	r

Now, the Sparse Graph and Dense Graph have been drawn below:

Sparse Graph:



Dense Graph:



4. a) Draw an undirected graph using the vertices y, p, z, x and r, where x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2. Also find the Depth First Search (DFS) sequence from the graph considering x is starting vertex

Solution:

Here, x = 70+5 = 75

y = 75+3 = 78

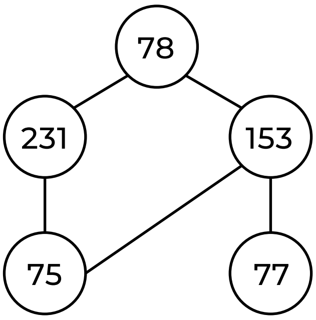
z = 75+78 = 153

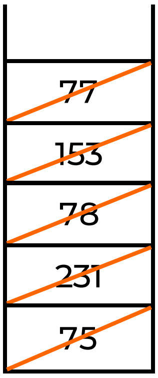
p = 78+153 = 231

r = 75+2 = 77

78	231	153	75	77
y	p	z	x	r

Now, the undirected graph has been drawn below:





Stack

Considering,
x (75) as starting vertex

DFS Result:

75, 231, 78, 153, 77

∴ Final DFS Sequence is: 75, 231, 78, 153, 77.

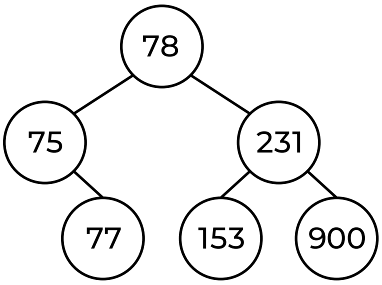
4. b) Construct a binary search tree (BST) using the nodes y, p, z, x, r and t, where x=last digit of your student id+5, y=x+3, z=x+y, p=y+z, r=x+2, t=900. Show the insertion and deletion of p+r and z, respectively in/from the BST.

Solution:

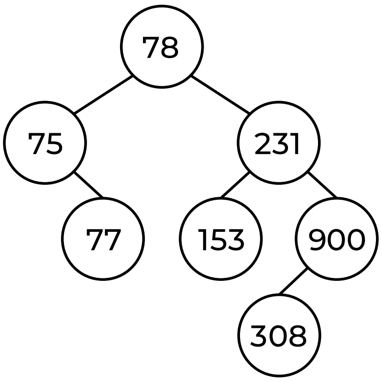
Here, x = 70+5 = 75   p = 78+153 = 231
y = 75+3 = 78   r = 75+2 = 77
z = 75+78 = 153   t = 900

78	231	153	75	77	900
y	p	z	x	r	t

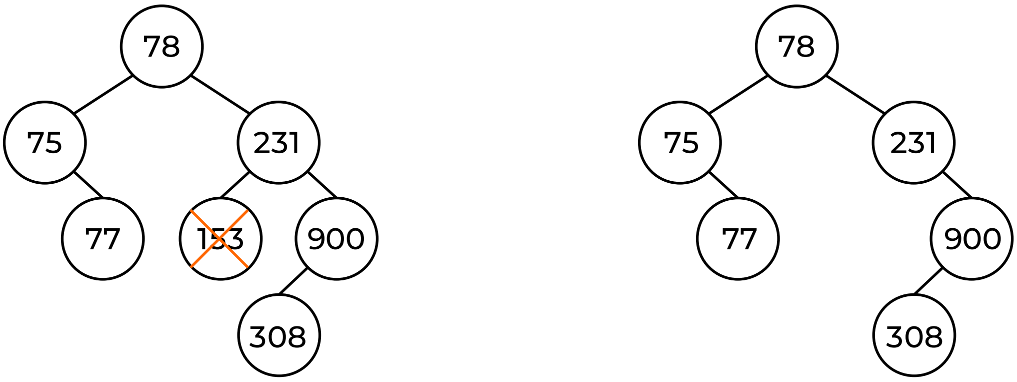
The binary search tree (BST) has been drawn below:



Inserting p+r: *(231+77 = 308)*



Deleting z: *(153)*



4. c) Write an algorithm to display all elements stored in QUEUE implemented by an array.

Solution:

Algorithm:

```
void displayQueue() {
    if (isEmpty()) {
        printf("Queue is empty!");
    }
    else {
        int i = front;
        while (i != rear);
            printf("%d ", arr[i]);
            i = i+1 % SIZE;
        }
    }
}
```

4. d) Which Data Structures are appropriate to implement the following and why?

i) Different areas of Dhaka City with distance
ii) Line in front of a Lift
iii) Binary Tree Construction

Solution:

i) Different areas of Dhaka City with a distance can be representing with Weighted Graph. The vertices will represent the areas name and edges weight will be represent the distance of both area.

ii) Line in front a lift can be represented with Queue. In line in front of a lift, which people came first he/she can go on the lift first, which is similar to Queue's First-In-First-Out policy.

iii) Array is ideal data structure to implementing Binary Tree. In array, we insert value of all the node of the tree level wisely and we to go left child, right child and parent of any nodes by formulas.