SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type:** Lab	**Academic Year:** 2025-26
Course Coordinator Name		Dr. S Vairachilai	
Instructor(s) Name			
Course Code	23CS302PC305	**Course Title**	Competitive Programming
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	Week I	**Time(s)**	NA
Duration	2 Hours each	**Applicable to Batches**	ALL batches
Assignment Number: Week 7 -FENWICK TREES			

Day	Question	Expected time to complete
Monday	**1_ Practical Exercises with Fenwick Trees -Binary Indexed Trees: Problem: Student Attendance Analysis Using Fenwick Tree** In a college, attendance of a student is recorded **daily** for a semester. Each day, the student earns **attendance points** (for example, based on presence in multiple sessions). Sometimes, attendance records are **corrected** due to late entries or verification. The college wants a fast system to: 1. **Update** attendance points of a particular day 2. **Find total attendance points** from Day 1 to a given day To perform these operations efficiently, a **Fenwick Tree (Binary Indexed Tree)** is used. **Task** Write a program using **Fenwick Tree** to support: • **Update operation**: Modify attendance points of a given day • **Query operation**: Find total attendance points till a given day **Example Test Case 1** **Input** • Number of days: **5** • Daily attendance points: **[1, 2, 1, 2, 1]**	**15th Feb, 2026, 5:00PM**

Operations:

1. Find total attendance points till Day **4**
2. Update Day **3** attendance from **1 → 2**
3. Find total attendance points till Day **4**

Output

• Total attendance till Day 4 = **6**

• After update, total attendance till Day 4 = **7 Explanation**

Initial attendance record:

Day: 1 2 3 4 5

Attendance: 1 2 1 2 1

Query 1

1 + 2 + 1 + 2 = 6

Update

Day 3 attendance corrected from **1 → 2**

Increase = **+1**

Query 2

1 + 2 + 2 + 2 = 7

2_Problem: Daily Sales Revenue Analysis Using Fenwick Tree

A retail store records its daily sales revenue for a month.
Each day, the total revenue earned is stored in the system.
Occasionally, sales figures are corrected due to billing errors, returned items, or late entries.
The store management requires a fast and efficient system to:

1. **Update the sales revenue of a particular day**
2. **Find the total sales revenue from Day 1 to a given day**

To perform these operations efficiently, a **Fenwick Tree (Binary Indexed Tree)** is used.

Task

Write a program using a **Fenwick Tree** to support the following operations:

- **Update Operation**
  Modify the sales revenue of a given day.
- **Query Operation**
  Find the total sales revenue from Day 1 up to a specified day.

Input Specification

- An integer **N** representing the number of days.
- An array **A[1…N]**, where A[i] denotes sales revenue on the *i-th* day.
- A set of operations:
  - Update (i, x) → Update sales revenue of Day i to value x
  - Query (d) → Find total sales revenue from Day 1 to Day d

Output Specification

- Display the total sales revenue for each query operation.

Example Test Case 1

Input

- Number of days: **7**
- Daily sales revenue:
  [200, 150, 300, 250, 100, 180, 220]

Operations:

1. Find total sales revenue till **Day 6**
2. Update **Day 5** revenue from **100 → 160**
3. Find total sales revenue till **Day 6**

Output

- Total sales revenue till Day 6 = **1180**
- After update, total sales revenue till Day 6 = **1240**

Explanation

Initial Sales Revenue Record

Day	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Revenue	200	150	300	250	100	180	220

Query 1:

Total sales revenue till Day 6

200 + 150 + 300 + 250 + 100 + 180 = 1180

Update:

Day 5 revenue corrected from **100 → 160**

Increase = **+60**

Query 2:

Total sales revenue till Day 6

200 + 150 + 300 + 250 + 160 + 180 = 1240

Tuesday	**1_ Practical Exercises with Fenwick Trees -Binary Indexed Trees: Problem: Rainfall Measurement Using Fenwick Tree** A meteorological department records the **daily rainfall (in mm)** for a region over several days. Sometimes, rainfall values are **corrected** due to sensor recalibration or delayed data updates. To efficiently manage rainfall data, the department uses a **Fenwick Tree (Binary Indexed Tree)** to: 1. **Update** the rainfall value of a specific day 2. **Query** the total rainfall from Day 1 to a given day **Task** Write a program using **Fenwick Tree** to support: • **Update operation**: Modify rainfall measurement of a given day • **Query operation**: Find cumulative rainfall till a given day **Example Test Case 1** **Input** • Number of days: **6** • Daily rainfall (mm): **[5, 12, 7, 10, 6, 8]** Operations: 1. Find total rainfall till Day **4** 2. Update Day **3** rainfall from **7 → 9** 3. Find total rainfall till Day **4** **Output** • Total rainfall till Day 4 = **34 mm** • After update, total rainfall till Day 4 = **36 mm** **Explanation** Initial rainfall data: Day: 1 2 3 4 5 6 Rain(mm): 5 12 7 10 6 8 **Query 1** 5 + 12 + 7 + 10 = 34 mm **Update** Rainfall on Day 3 corrected from **7 → 9** Increase = **+2 mm** **Query 2** 5 + 12 + 9 + 10 = 36 mm **2_Problem: Online Course Watch-Time Analysis Using Fenwick Tree** **Problem Statement** An online learning platform tracks the daily watch time (in minutes) of a student for a course. Due to buffering issues or session corrections, watch-time data for some days may be updated. The platform needs a fast system to: 1. **Update watch time of a particular day** 2. **Find total watch time from Day 1 to a given day**	16th Feb, 2025 5:00PM

A **Fenwick Tree (Binary Indexed Tree)** is used to efficiently manage this data.

Task

Write a program using a **Fenwick Tree** to support:

- **Update Operation**
  Modify watch time of a given day.
- **Query Operation**
  Find total watch time from Day 1 up to a specified day.

Input Specification

- Integer **N** – number of days
- Array **A[1…N]** – daily watch time in minutes
- Operations:
  - Update (i, x) → Update watch time on Day i
  - Query (d) → Total watch time from Day 1 to Day d

Example Test Case

Input

- Number of days: **5**
- Daily watch time (minutes): [30, 40, 20, 50, 10]

Operations

1. Query watch time till Day 4
2. Update Day 2 watch time from 40 → 55
3. Query watch time till Day 4

Output

- Total watch time till Day 4 = **140**
- After update, total watch time till Day 4 = **155**

Explanation

Initial:
30 + 40 + 20 + 50 = 140
After update (+15 on Day 2):
30 + 55 + 20 + 50 = 155

Wednesday	**WEEK7_3_ Practical Exercises with Fenwick Trees -Binary Indexed Trees:Problem: Hospital Patient Count Monitoring Using Fenwick Tree** A hospital records the **number of patients admitted each day** in a particular ward. Occasionally, patient counts are **updated** due to late admissions, discharges, or data corrections. To efficiently manage daily patient data, the hospital uses a **Fenwick Tree (Binary Indexed Tree)** that allows: 1. **Updating** the patient count for a specific day 2. **Querying** the total number of patients admitted from Day 1 to a given day **Task** Write a program using a **Fenwick Tree** to support: • **Update operation**: Modify the patient count of a given day • **Query operation**: Find cumulative patient count till a given day **Example Test Case 1** **Input** • Number of days: **7** • Daily patient admissions: **[18, 22, 20, 25, 19, 23, 21]** Operations: 1. Find total patient count till Day **5** 2. Update Day **4** patient count from **25 → 27** 3. Find total patient count till Day **5** **Output** • Total patients till Day 5 = **104** • After update, total patients till Day 5 = **106** **Explanation** Initial patient admission data: Day: 1 2 3 4 5 6 7 Patients:18 22 20 25 19 23 21 **Query 1** 18 + 22 + 20 + 25 + 19 = 104 **Update** Day 4 patient count corrected from **25 → 27** Increase = **+2 patients** **Query 2** 18 + 22 + 20 + 27 + 19 = 106 **Problem: Daily Step Count Analysis Using Fenwick Tree** **Problem Statement** A fitness tracking application records the number of steps taken by a user every day. Each day's step count is stored in the system. Sometimes, step counts are updated due to device synchronization issues or manual corrections. The application requires a fast system to:	**17th Feb, 2025 5:00PM**

1. **Update the step count of a particular day**
2. **Find the total number of steps from Day 1 to a given day**

To perform these operations efficiently, a **Fenwick Tree (Binary Indexed Tree)** is used.

Task

Write a program using a **Fenwick Tree** to support the following operations:

- **Update Operation**
  Modify the step count recorded on a given day.
- **Query Operation**
  Find the total number of steps from Day 1 up to a specified day.

Input Specification

- Integer **N** – number of days
- Array **A[1…N]** – number of steps taken each day
- Operations:
  - Update (i, x) → Update step count of Day i to x
  - Query (d) → Find total steps from Day 1 to Day d

Output Specification

- Display the total number of steps for each query operation.

Example Test Case

Input

- Number of days: **7**
- Daily step counts:
  [4500, 6000, 5200, 7000, 4800, 6500, 5000]

Operations:

1. Find total steps till **Day 5**
2. Update **Day 3** step count from **5200 → 5800**
3. Find total steps till **Day 5**

Output

- Total steps till Day 5 = **27,500**
- After update, total steps till Day 5 = **28,100**

Explanation

Initial Step Count Record

Day	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Steps	4500	6000	5200	7000	4800	6500	5000

Query 1:

Total steps till Day 5

4500 + 6000 + 5200 + 7000 + 4800 = 27,500

Update:

Day 3 steps corrected from **5200 → 5800**

Increase = **+600**

Query 2:

Total steps till Day 5

4500 + 6000 + 5800 + 7000 + 4800 = 28,100

Thursday	**WEEK7_4_ Practical Exercises with Fenwick Trees -Binary Indexed Trees:Problem: Library Book Borrowing Records :** **Problem Statement :** A university library records the number of books borrowed each day. Due to late returns or corrections, daily records may change. You are required to efficiently support: 1. Prefix Query – Find the total number of books borrowed from Day 1 to Day *x* 2. Update Operation – Update the number of books borrowed on a given day Implement a Binary Indexed Tree (Fenwick Tree) to process these operations in O(log n) time. **Input Format** The first line contains an integer T, the number of test cases. For each test case: • The first line contains an integer N, the number of days • The second line contains N space-separated integers, representing books borrowed each day • The third line contains an integer Q, the number of queries • The next Q lines contain queries of the form: ◦ SUM x → Find total books borrowed till Day x ◦ UPDATE i val → Increase books borrowed on Day i by val **Output Format** For each SUM query, print the result on a new line. **Constraints** • $1 \le T \le 20$ • $1 \le N \le 200000$ • $-10^9 \le arr[i] \le 10^9$ • $1 \le Q \le 200000$ • $0 \le i < N$ **Sample Input** 1 6 12 15 10 20 18 25 4 SUM 4 UPDATE 3 5 SUM 4	**18th Feb, 2025 5:00PM**

SUM 6

Sample Output

57
62
105

Explanation

Initial array:
[12, 15, 10, 20, 18, 25]

- SUM 4 → 12 + 15 + 10 + 20 = 57
- UPDATE 3 5 → arr[3] = 20 + 5 = 25
- Updated array: [12, 15, 10, 25, 18, 25]
- SUM 4 → 12 + 15 + 10 + 25 = 62
- SUM 6 → Total = 105

Problem: Daily Water Consumption Analysis Using Fenwick Tree

A smart water management system records the daily water consumption (in liters) of a household.
Each day's water usage is stored in the system.
Due to meter recalibration or delayed readings, water consumption values for some days may be updated.
The system requires a fast and efficient method to:

1. **Update water consumption of a particular day**
2. **Find the total water consumption from Day 1 to a given day**

To perform these operations efficiently, a **Fenwick Tree (Binary Indexed Tree)** is used.

Task

Write a program using a **Fenwick Tree** to support the following operations:

- **Update Operation**
  Modify the water consumption value of a given day.
- **Query Operation**
  Find the total water consumption from Day 1 up to a specified day.

Input Specification

- Integer **N** – number of days
- Array **A[1…N]** – daily water consumption in liters
- Operations:
  - Update (i, x) → Update water consumption on Day i to value x
  - Query (d) → Find total water consumption from Day 1 to Day d

Output Specification

- Display the total water consumption for each query operation.

Example Test Case

Input

- Number of days: **6**
- Daily water consumption (liters):
  [120, 135, 110, 150, 140, 125]

Operations:

1. Find total water consumption till **Day 4**
2. Update **Day 2** consumption from **135 → 145**
3. Find total water consumption till **Day 4**

Output

- Total water consumption till Day 4 = **515**
- After update, total water consumption till Day 4 = **525**

Explanation

Initial Water Consumption Record

Day	**1**	**2**	**3**	**4**	**5**	**6**
Consumption (L)	120	135	110	150	140	125

Query 1:

Total water consumption till Day 4

120 + 135 + 110 + 150 = 515

Update:

Day 2 consumption corrected from **135 → 145**

Increase = **+10**

Query 2:

Total water consumption till Day 4

120 + 145 + 110 + 150 = 525

Friday	**WEEK7_5_ Practical Exercises with Fenwick Trees -Binary Indexed Trees:Problem: Monthly Electricity Consumption Tracking** **Problem Statement** An electricity board records daily power consumption (in units) for a locality. Consumption values may change due to meter corrections. You must efficiently support: • Prefix Sum Queries for total power consumption up to a given day • Update Operations when daily readings change Use a Fenwick Tree to process queries efficiently. **Input Format** • Integer T – number of test cases For each test case: • Integer N – number of days • Array of N integers – power units consumed per day • Integer Q – number of operations • Next Q lines: ◦ SUM x ◦ UPDATE i val **Sample Input** 1 7 30 28 35 40 33 38 36 3 SUM 5 UPDATE 4 -3 SUM 5 **Sample Output** **166** **163** **Explanation** Initial data: [30, 28, 35, 40, 33, 38, 36] • SUM 5 → 30 + 28 + 35 + 40 + 33 = 166 • UPDATE 4 -3 → arr[4] = 33 − 3 = 30 • SUM 5 → 163 **Problem: Daily Mobile Data Usage Analysis Using Fenwick Tree** A mobile service provider records the daily mobile data usage (in MB) of a user. Each day's data consumption is logged in the system. Sometimes, usage records are corrected due to synchronization issues or billing adjustments. The service provider needs a fast and efficient system to: 1. **Update mobile data usage of a particular day**	19th Feb, 2025 5:00PM

2. **Find the total mobile data usage from Day 1 to a given day**

To perform these operations efficiently, a **Fenwick Tree (Binary Indexed Tree)** is used.

Task

Write a program using a **Fenwick Tree** to support the following operations:

- **Update Operation**
  Modify the mobile data usage of a given day.
- **Query Operation**
  Find the total mobile data usage from Day 1 up to a specified day.

Input Specification

- Integer **N** – number of days
- Array **A[1…N]** – daily mobile data usage in MB
- Operations:
  - Update (i, x) → Update data usage on Day i to value x
  - Query (d) → Find total data usage from Day 1 to Day d

Output Specification

- Display the total mobile data usage for each query operation.

Example Test Case

Input

- Number of days: **7**
- Daily mobile data usage (MB):
  [500, 650, 400, 800, 550, 700, 600]

Operations:

1. Find total mobile data usage till **Day 6**
2. Update **Day 3** data usage from **400 → 480**
3. Find total mobile data usage till **Day 6**

Output

- Total mobile data usage till Day 6 = **3600 MB**
- After update, total mobile data usage till Day 6 = **3680 MB**

Explanation

Initial Data Usage Record

Day	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Usage (MB)	500	650	400	800	550	700	600

Query 1:

Total data usage till Day 6

500 + 650 + 400 + 800 + 550 + 700 = 3600 MB

Update:

Day 3 data usage corrected from **400 → 480**

Increase = **+80 MB**

	Query 2: Total data usage till Day 6 500 + 650 + 480 + 800 + 550 + 700 = 3680 MB	