Competitive Programming

Assignment-04

2303A51463

B-07

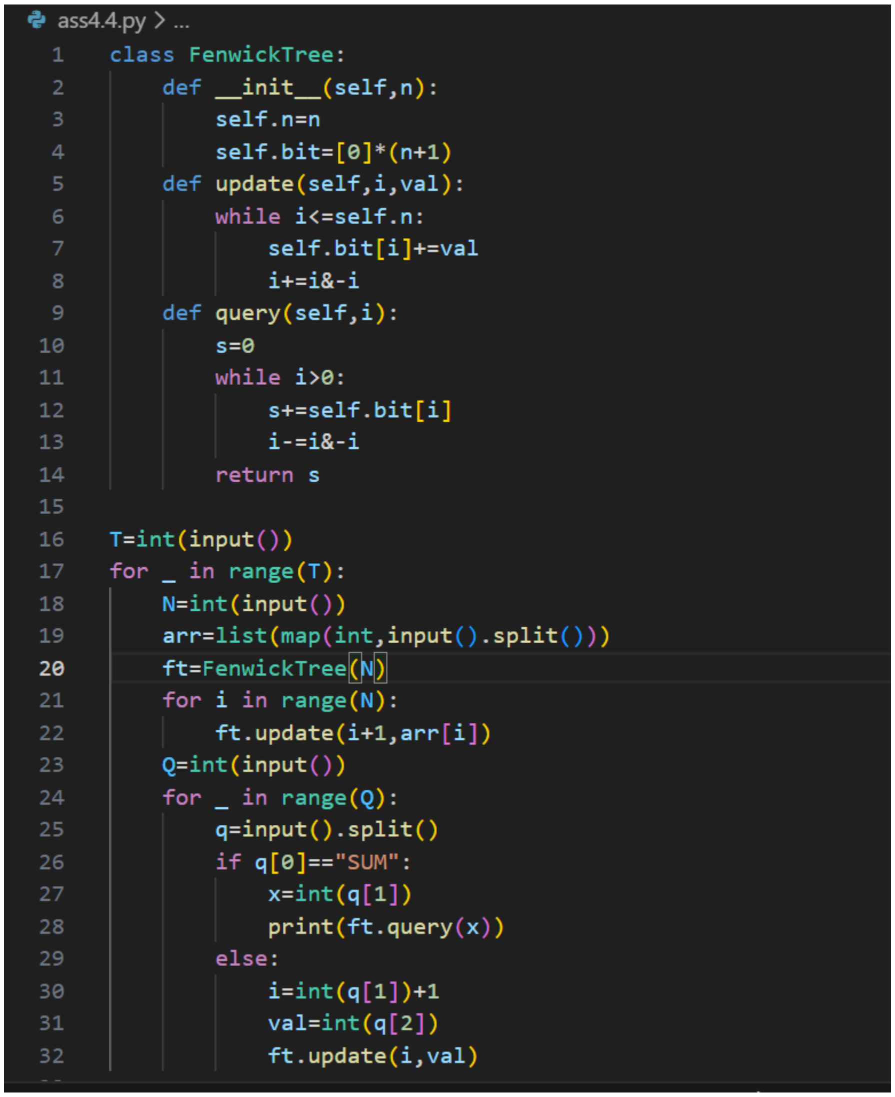
WEEK7_4: Practical Exercises with Fenwick Trees –Binary Indexed Trees

Problem: Library Book Borrowing Records:

Algorithm:

1. Read integer **T** (number of test cases).
2. For each test case:
    a. Read integer **N** (number of days).
    b. Read array **A[1...N]** representing books borrowed each day.
    c. Initialize a Fenwick Tree **BIT[1...N]** with all values as 0.
    d. Build the Fenwick Tree:
        i. For `i = 1 to N`, perform `update(i, A[i])`.
    e. Read integer **Q** (number of queries).
    f. For each query:
        i. If query is `SUM x`:
            1. Compute prefix sum from Day 1 to Day x using Fenwick Tree.
            2. Print the result.
        ii. If query is `UPDATE i val`:
            1. Increase books borrowed on Day `i` by `val`.
            2. Perform `update(i+1, val)` in Fenwick Tree.
3. End.

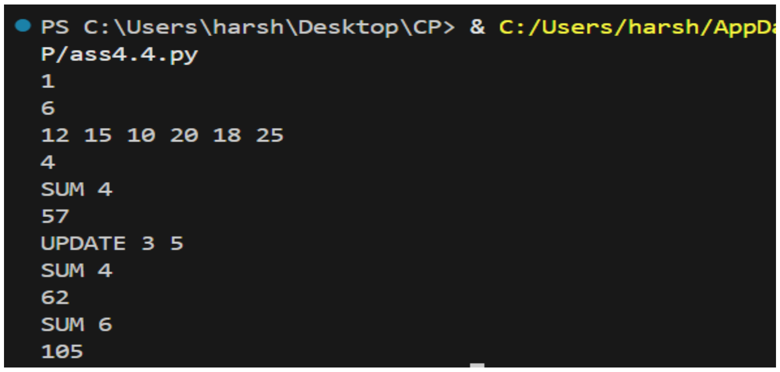
Code :



```
ass4.4.py > ...
class FenwickTree:
    def __init__(self,n):
        self.n=n
        self.bit=[0]*(n+1)
    def update(self,i,val):
        while i<=self.n:
            self.bit[i]+=val
            i+=i&-i
    def query(self,i):
        s=0
        while i>0:
            s+=self.bit[i]
            i-=i&-i
        return s

T=int(input())
for _ in range(T):
    N=int(input())
    arr=list(map(int,input().split()))
    ft=FenwickTree(N)
    for i in range(N):
        ft.update(i+1,arr[i])
    Q=int(input())
    for _ in range(Q):
        q=input().split()
        if q[0]=="SUM":
            x=int(q[1])
            print(ft.query(x))
        else:
            i=int(q[1])+1
            val=int(q[2])
            ft.update(i,val)
```

Output:



```
PS C:\Users\harsh\Desktop\CP> & C:/Users/harsh/AppDa
P/ass4.4.py
1
6
12 15 10 20 18 25
4
SUM 4
57
UPDATE 3 5
SUM 4
62
SUM 6
105
```

Problem: Daily Water Consumption Analysis Using Fenwick Tree

Algorithm:

1. Read integer **N** (number of days).

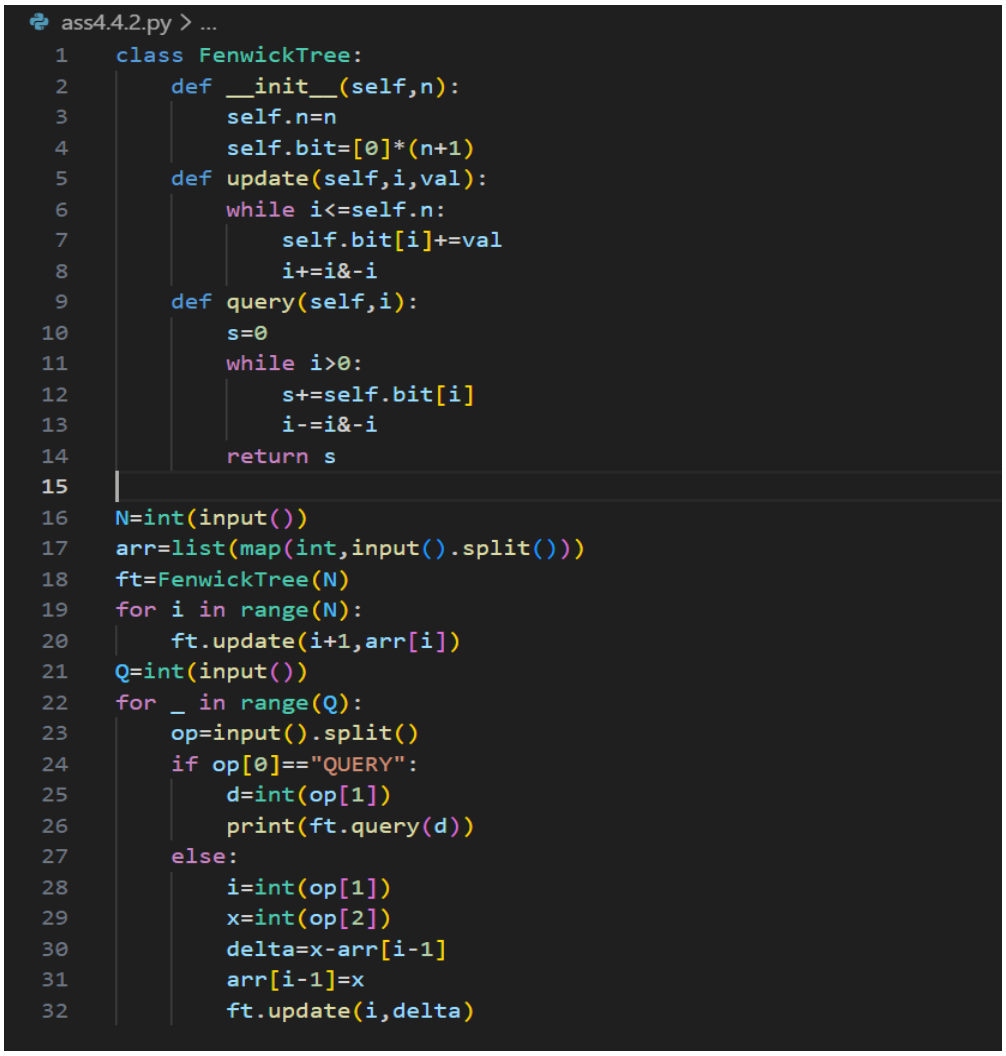
2. Read array **A[1...N]** representing daily water consumption.

3. Initialize a Fenwick Tree **BIT[1...N]** and build it using the array values.

4. Read integer **Q** (number of operations).

5. For each operation:
- If operation is **QUERY d**, print total water consumption from Day 1 to Day d.
- If operation is **UPDATE i x**, update water consumption on Day `i` and modify Fenwick Tree.

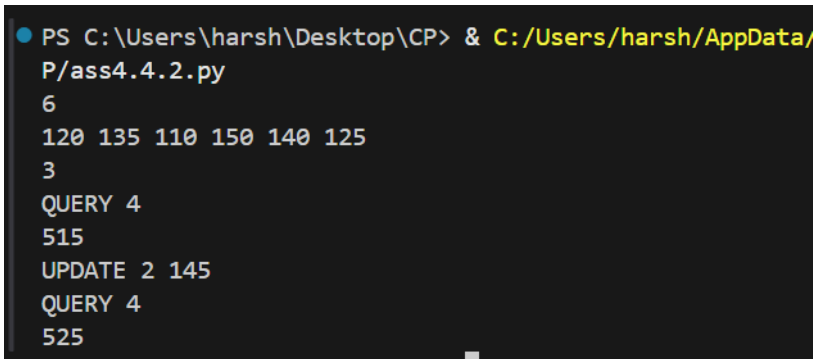
6. End.


ass4.4.2.py > ...

```python
class FenwickTree:
    def __init__(self,n):
        self.n=n
        self.bit=[0]*(n+1)
    def update(self,i,val):
        while i<=self.n:
            self.bit[i]+=val
            i+=i&-i
    def query(self,i):
        s=0
        while i>0:
            s+=self.bit[i]
            i-=i&-i
        return s

N=int(input())
arr=list(map(int,input().split()))
ft=FenwickTree(N)
for i in range(N):
    ft.update(i+1,arr[i])
Q=int(input())
for _ in range(Q):
    op=input().split()
    if op[0]=="QUERY":
        d=int(op[1])
        print(ft.query(d))
    else:
        i=int(op[1])
        x=int(op[2])
        delta=x-arr[i-1]
        arr[i-1]=x
        ft.update(i,delta)
```

Output:



```
PS C:\Users\harsh\Desktop\CP> & C:/Users/harsh/AppData/
P/ass4.4.2.py
6
120 135 110 150 140 125
3
QUERY 4
515
UPDATE 2 145
QUERY 4
525
```

WEEK7_5_ Practical Exercises with Fenwick Trees –Binary Indexed Trees

Problem: Monthly Electricity Consumption Tracking

Algorithm:

1. Read integer **T** (number of test cases).

2. For each test case:
  a. Read integer **N** (number of days).
  b. Read array **A[1...N]** representing daily electricity consumption.

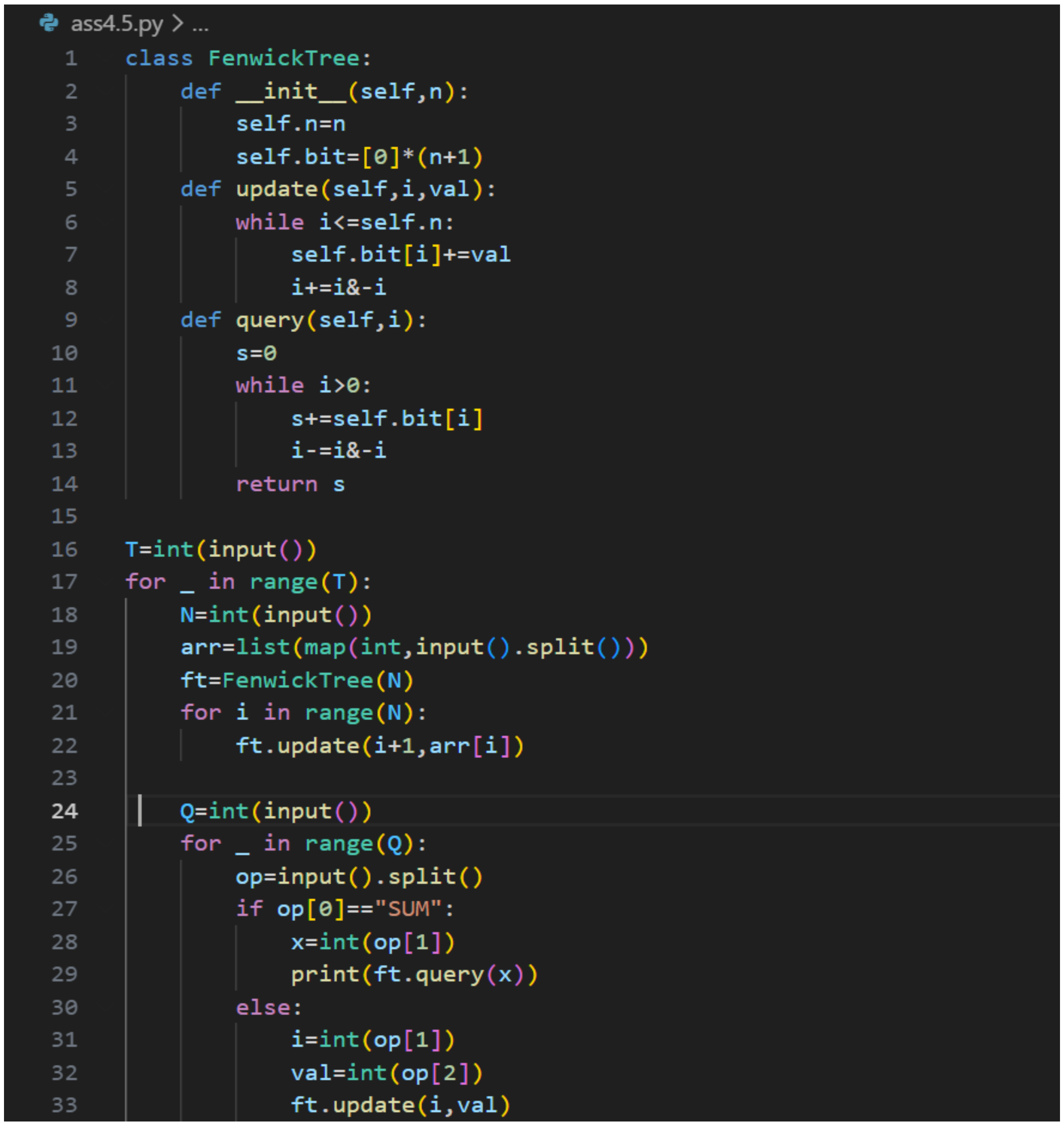
3. Initialize a Fenwick Tree **BIT[1...N]** with all values as 0.

4. Build the Fenwick Tree by performing `update(i, A[i])` for `i = 1` to N.

5. Read integer **Q** (number of operations).

6. For each operation:
  • If operation is **SUM x**, compute and print prefix sum from Day 1 to Day x.
  • If operation is **UPDATE i val**, increase the value on Day `i` by `val` and update Fenwick Tree.

7. End.

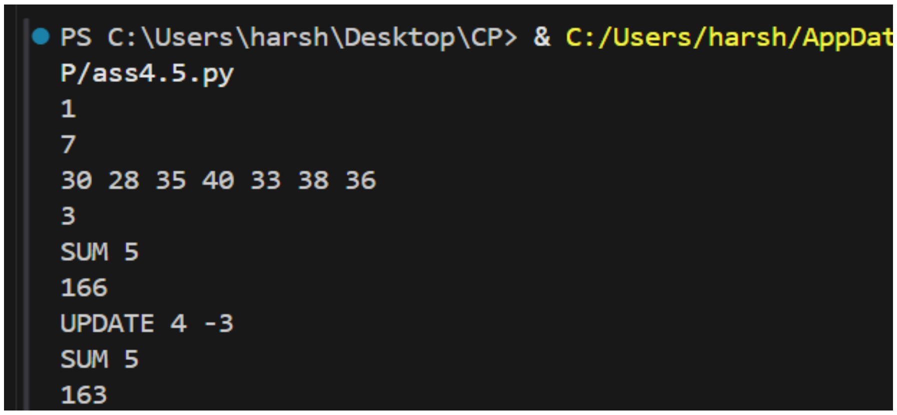


```
ass4.5.py > ...
class FenwickTree:
    def __init__(self,n):
        self.n=n
        self.bit=[0]*(n+1)
    def update(self,i,val):
        while i<=self.n:
            self.bit[i]+=val
            i+=i&-i
    def query(self,i):
        s=0
        while i>0:
            s+=self.bit[i]
            i-=i&-i
        return s

T=int(input())
for _ in range(T):
    N=int(input())
    arr=list(map(int,input().split()))
    ft=FenwickTree(N)
    for i in range(N):
        ft.update(i+1,arr[i])

    Q=int(input())
    for _ in range(Q):
        op=input().split()
        if op[0]=="SUM":
            x=int(op[1])
            print(ft.query(x))
        else:
            i=int(op[1])
            val=int(op[2])
            ft.update(i,val)
```

Output:



```
PS C:\Users\harsh\Desktop\CP> & C:/Users/harsh/AppDat
P/ass4.5.py
1
7
30 28 35 40 33 38 36
3
SUM 5
166
UPDATE 4 -3
SUM 5
163
```

Problem: Daily Mobile Data Usage Analysis Using Fenwick Tree

Algorithm:

1. Read integer **N** (number of days).

2. Read array **A[1…N]** representing daily mobile data usage in MB.

3. Initialize a Fenwick Tree **BIT[1…N]** with all values as 0.

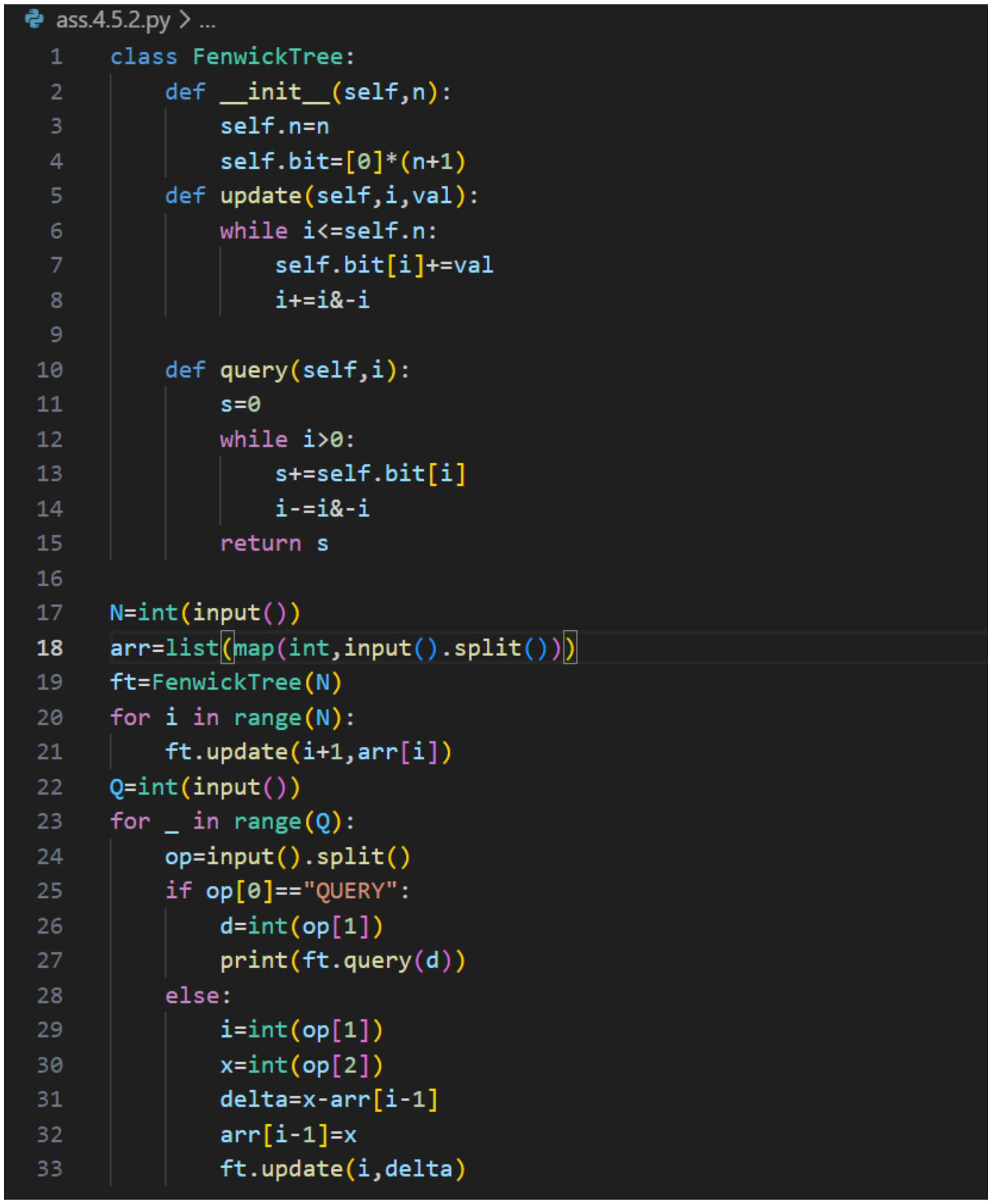
4. Build the Fenwick Tree by performing `update(i, A[i])` for `i = 1` to `N`.

5. Read integer **Q** (number of operations).

6. For each operation:

- If operation is **QUERY d**, compute and print total mobile data usage from Day 1 to Day d.
- If operation is **UPDATE i x**,
  – Compute `delta = x - A[i]`
  – Update `A[i] = x`
  – Perform `update(i, delta)` in Fenwick Tree.

7. End.

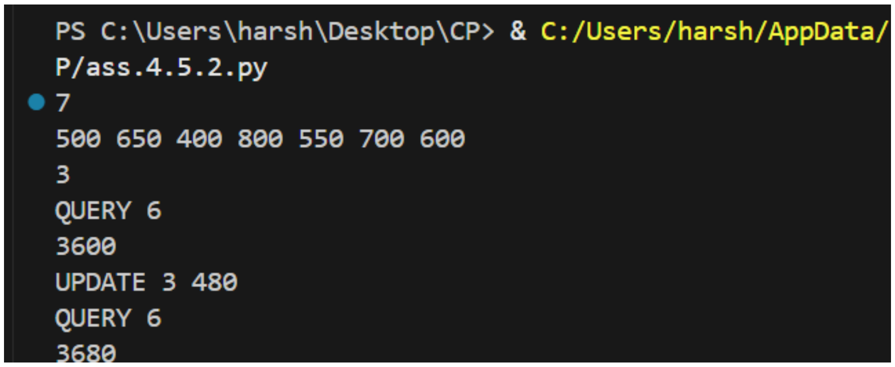


```
ass.4.5.2.py > ...
class FenwickTree:
    def __init__(self,n):
        self.n=n
        self.bit=[0]*(n+1)
    def update(self,i,val):
        while i<=self.n:
            self.bit[i]+=val
            i+=i&-i

    def query(self,i):
        s=0
        while i>0:
            s+=self.bit[i]
            i-=i&-i
        return s

N=int(input())
arr=list(map(int,input().split()))
ft=FenwickTree(N)
for i in range(N):
    ft.update(i+1,arr[i])
Q=int(input())
for _ in range(Q):
    op=input().split()
    if op[0]=="QUERY":
        d=int(op[1])
        print(ft.query(d))
    else:
        i=int(op[1])
        x=int(op[2])
        delta=x-arr[i-1]
        arr[i-1]=x
        ft.update(i,delta)
```

Output:



```
PS C:\Users\harsh\Desktop\CP> & C:/Users/harsh/AppData/
P/ass.4.5.2.py
7
500 650 400 800 550 700 600
3
QUERY 6
3600
UPDATE 3 480
QUERY 6
3680
```