Assignment- 4.7

HT-No: 2303A51963

Name: k.prudhvi raj

Batch: 24

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 the following operations:

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 the number of 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.

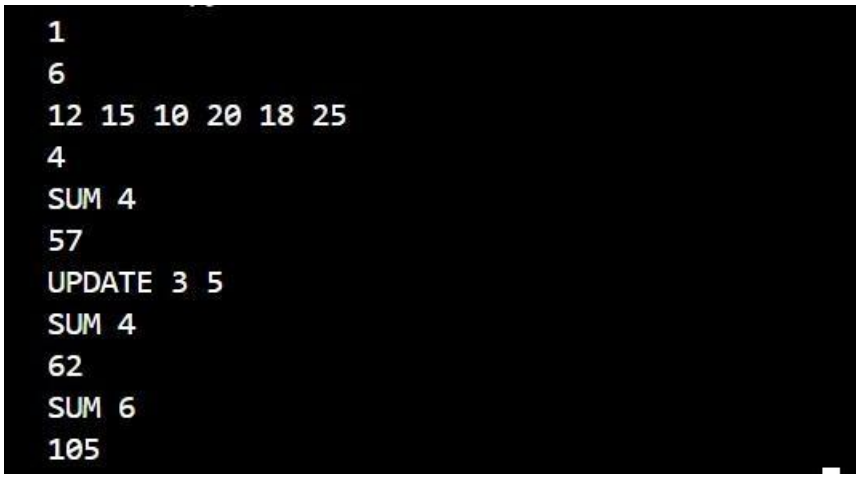
Constraints:

- $1 \le T \le 20$
- $1 \le N \le 200000$
- $-10^9 \le arr[i] \le 10^9$ • $1 \le Q \le 200000$ **PYTHON CODE:**

```
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:  # UPDATE
          i = int(q[1])
val = int(q[2])
ft.update(i, val)
```

Output:



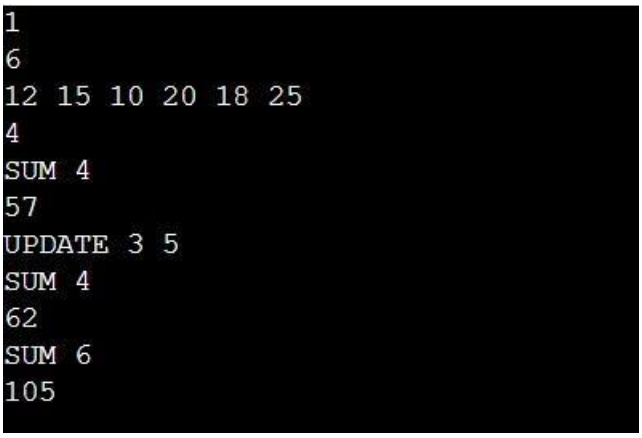
C CODE:

```
#include <stdio.h>
#include <string.h> #define
MAXN 200005 long long
BIT[MAXN];
int N; void update(int i, long
long val) {     while (i <= N) {
BIT[i] += val;        i += i & (-i);
   }
} long long query(int i) {
long long sum = 0;
while (i > 0) {        sum
+= BIT[i];
      i -= i & (-i);
   }
   return sum;
} int main() {
   int T;
   scanf("%d", &T);     while (T--) {        scanf("%d",
&N);        for (int i = 1; i <=
N; i++)           BIT[i] = 0;       for (int i =
1; i <= N; i++) {              long long x;
scanf("%lld", &x);            update(i, x);
}          int Q;          scanf("%d", &Q);
while (Q--) {               char type[10];
scanf("%s",  type);                        if
(strcmp(type, "SUM") == 0) {
```

```
            int x;            scanf("%d",
&x);            printf("%lld\n",
query(x));
        } else {            int i;
long long val;            scanf("%d
%lld", &i, &val);            update(i,
val);

        }
}   }
return 0;
}
```

Output:



JAVA CODE:

```
import java.io.*; import
java.util.*;


public class Main {
  static int N;    static
long[] BIT;
```

```
   static void update(int i, long val){
while(i<=N){            BIT[i]+=val;
i+=i&(-i);
      }
   }

   static long query(int i){        long
sum=0;        while(i>0){
sum+=BIT[i];            i-=i&(-i);
      }        return
sum;
   }

   public static void main(String[] args)throws Exception{
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));        StringTokenizer st;        int
T=Integer.parseInt(br.readLine());        while(T-->0){
         N=Integer.parseInt(br.readLine());
BIT=new long[N+1];            st=new
StringTokenizer(br.readLine());            for(int i=1;i<=N;i++){
update(i,Long.parseLong(st.nextToken()));
         }
         int Q=Integer.parseInt(br.readLine());
while(Q-->0){               st=new
StringTokenizer(br.readLine());               String
type=st.nextToken();
if(type.equals("SUM")){                        int
x=Integer.parseInt(st.nextToken());
```

```
            System.out.println(query(x));
        }else{              int
i=Integer.parseInt(st.nextToken());             long
val=Long.parseLong(st.nextToken());
            update(i,val);
        }
      }
    }
  }
}
```

Output: