Summenbäume

Gegeben ist ein Array mit N Zahlen, nummeriert von 1 bist N. Am Anfan sind alle diese Zahlen 0. Du sollst zwei Operationen auf diesem Array ausführen können.

- `update(i, v)` du musst die Zahl **v** zu dem Element mit Index **i** addieren.
- `query(p, q)` du musst die Summe aller Elemente in Intervall [p, q] berechnen.

Aufgabenstellung:

Du musst in einem File die Funktionen **init(N), update(i, v) und query(p, q)** implementieren.
init(N) … wird am Beginn der Ausführung genau einmal aufgerufen.
update(i, v) … soll die oben beschriebenen Auswirkungen auf das Array haben.
query(p, q) … muss den Wert der oben beschriebenen Abfrage zurückgeben.

Implementierungsdetails:

Du musst genau eine Datei einreichen. Diese Datei muss BIT.c, BIT.cpp benannt sein. Diese Datei muss die oben beschriebenen Funktionen implementieren und die folgenden Signaturen respektieren:

```
void init(int N);
void update(int i, int v);
long long query(int p, int q);
```

Beispielgrader
Der Beispielgrader liest die Eingabe im folgenden Format:

Zeile 1: die Zahl N und die Anzahl der Kommandos Q.
In jeder der folgenden Q Zeilen befindet sich entweder:

- 0 gefolgt von einem Leerzeichen und zwei durch Leerzeichen getrennte Integer, i und v – entspricht einem update.
- 1 gefolgt von zwei durch Leerzeichen getrente Integer p, q – entspricht einem query.

Der Beispielgrader wird die, vom query Befehl zurückgegebenen Zahlen ausgeben.
Eine Zahl pro Zeile.

Beispiel:

`init(8)`	`[0, 0, 0, 0, 0, 0, 0, 0 ]`
`update(4, 26)`	`[0, 0, 0, 26, 0, 0, 0, 0]`
`update(5, 80)`	`[0, 0, 0, 26, 80, 0, 0, 0]`
`update(7, 20)`	`[0, 0, 0, 26, 80, 0, 20, 0]`
`query(8, 8)`	`Ergebnis: 0`
`update(1, 14)`	`[14, 0, 0, 26, 80, 0, 20, 0]`
`Query(1, 6)`	`Ergebnis: 120`

Grading:

Für alle Subtasks gilt folgende Annahme: **|v| <= 10 000 000.**

Subtask 1(20 P):
Du kannst annehmen, dass **1** <= **N** <= **1000** und **1** <= **Q** <= **1000** gilt.

Subtask 1(80 P):
Du kannst annehmen, dass **1** <= **N** <= **100 000** und **1** <= **Q** <= **100 000** gilt.