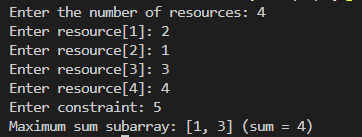
|  |  |
| --- | --- |
| NAME | Rohit Jangid |
| SECTION | A2-B4 |
| ROLL NO. | 56 |

PRACT-4

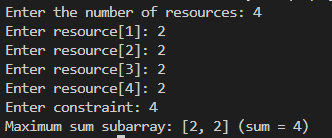
1. Basic small array

* resources = [2, 1, 3, 4], constraint = 5
* Best subarray: [2, 1] or [1, 3] → sum = 4
* Checks simple working.



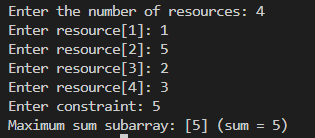
2. Exact match to constraint

* resources = [2, 2, 2, 2], constraint = 4
* Best subarray: [2, 2] → sum = 4
* Tests exact utilization.



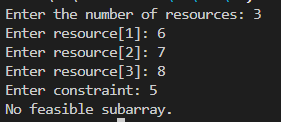
3. Single element equals constraint

* resources = [1, 5, 2, 3], constraint = 5
* Best subarray: [5] → sum = 5
* Tests one-element solution.



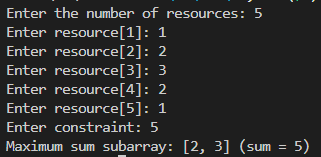
4. All elements smaller but no combination fits

* resources = [6, 7, 8], constraint = 5
* No feasible subarray.
* Tests &quot;no solution&quot; case.



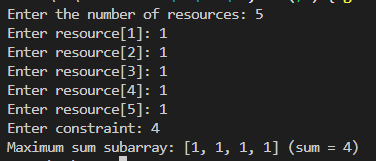
5. Multiple optimal subarrays

* resources = [1, 2, 3, 2, 1], constraint = 5
* Best subarrays: [2, 3] and [3, 2] → sum = 5
* Tests tie-breaking (should return either valid subarray).



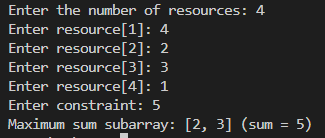
6. Large window valid

* resources = [1, 1, 1, 1, 1], constraint = 4
* Best subarray: [1, 1, 1, 1] → sum = 4
* Ensures long window works.



7. Sliding window shrink needed

* resources = [4, 2, 3, 1], constraint = 5
* Start [4,2] = 6 (too big) → shrink to [2,3] = 5.
* Tests dynamic window adjustment.



8. Empty array

* resources = [], constraint = 10
* Output: no subarray.
* Edge case: empty input.



9. Constraint = 0

* resources = [1, 2, 3], constraint = 0
* No subarray possible.
* Edge case: zero constraint.

