Name**: Somya Mehta**

Roll No: 190001058

**//Assignment-5 \_190001058.cpp**

**Code Logic:-**

I have used Binary Search here to calculate the minimum time it would take for the painters to complete their work.We are able to use  **binary Search** here because if a time t is sufficient for the workers to complete their task then any value of time greater than t will also work hence we can reduce our search space by applying binary search here.For a particular time t we can check that whether the task can be finished by the workers in the given time frame or not in O(n).Hence the time Complexity is **O(nlog(sum of elements in the array))**

**Source Code:-**

*//Somya Mehta*

*//190001058*

*//Assignment-5*

*#include* <bits/stdc++.h>

typedef long long ll;

typedef long double ld;

*#define* fr(i, a, b) *for* (ll i = a; i < b; i++)

*#define* rf(i, a, b) *for* (ll i = a; i >= b; i--)

typedef std::vector<long long> vi;

*#define* F first

*#define* S second

*#define* fast                      \

    ios\_base::sync\_with\_stdio(0); \

    cin.tie(0);                   \

    cout.tie(0);

*#define* mod 1000000007

*#define* PB push\_back

*#define* MP make\_pair

*#define* PI 3.14159265358979323846

*#define* all(a) a.begin(), a.end()

*#define* mx(a) \*max\_element(all(a))

*#define* mn(a) \*min\_element(all(a))

const ll INF = LLONG\_MAX / 2;

const ll N = 2e5 + 1;

using namespace std;

int arr[100005], r = 0, n, k;

bool check(int x)

{

    int sum = 0, number\_of\_workers\_used = 1;

*for* (int i = 0; i < n; i++)

    {

*if* (arr[i] + sum <= x)

            sum += arr[i];

*else*

        {

            number\_of\_workers\_used++;

            sum = 0;

            sum += arr[i];

        }

    }

*if* (number\_of\_workers\_used > k)

*return* false;

*return* true;

}

int main()

{

    cin >> n >> k;

    int l = 0;

*for* (int i = 0; i < n; i++)

    {

        cin >> arr[i];

        l = max(arr[i], l);

        r += arr[i];

    }

*while* (l <= r)

    {

        int mid = (l + r) / 2;

*if* (check(mid))

        {

            r = mid - 1;

        }

*else*

        {

            l = mid + 1;

        }

    }

    cout << r + 1 << "\n";

}

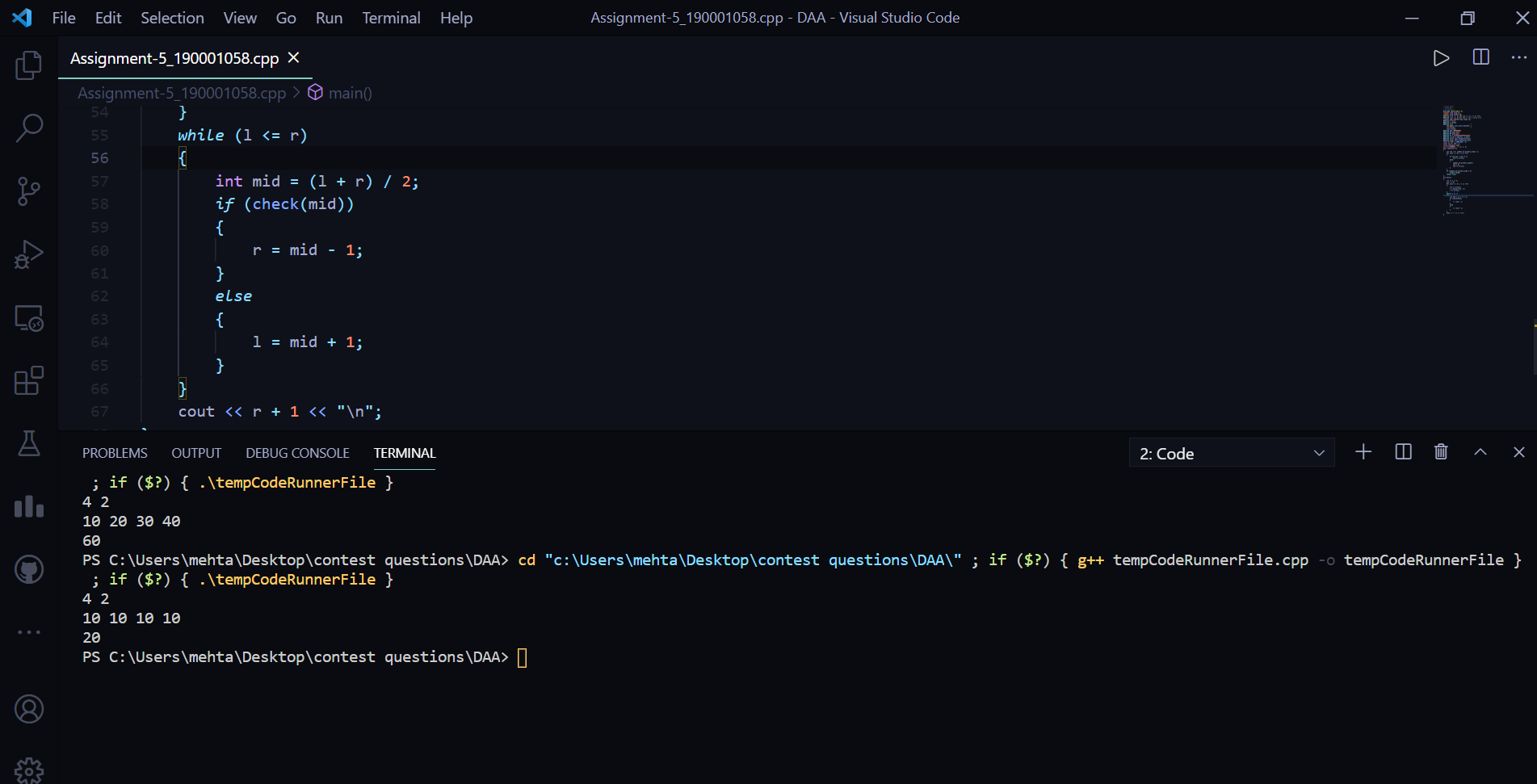
**Input-OUTPUT ScreenShots:-**

Input : k = 2, A = {10, 20, 30, 40}

Output : Shown Below

Input : k = 2, A = {10, 10, 10, 10}

Output : Shown Below.



-----------------------x-----------------------------------x---------------------------------------