3.c. 3-G-Burger Problem

Aim:

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories.
If he has eaten *i* burgers with c calories each, then he has to run at least $3^i * c$ kilometers to burn out the calories. For example, if he ate 3 burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$.
But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

Input Format

First Line contains the number of burgers
Second line contains calories of each burger which is n space-separate integers

Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

Sample Input

```
3
5 10 7
```

Sample Output

```
76
```

Algorithm:

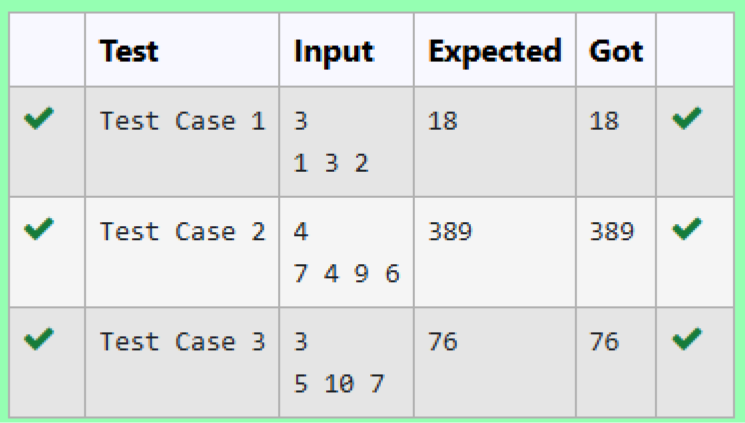
```
int main() {
    initialize n // number of elements
    read n from user

    initialize cal array of size n // array to hold integers

    // read values into the cal array
    for i from 0 to n-1 {
        read cal[i] from user
    }

    // sorting the array using bubble sort
    for i from 0 to n–2 {
        for j from 0 to n–i–2 {
            if call[j] is greater than call[j+1] {
                // swap call[j] and call[j+1]
                initialize temp as call[j]
                call[j] = call[j+1]
                call[j+1] = temp
            }
        }
    }

    initialize mulfact // variable to hold power value initialize
    sum to 0 // variable to hold the final sum initialize h to
    n-1 // index for the last element

    // compute the weighted sum
    for i from 0 to n–1 {
        mulfact = n raised to the power of i // compute nˆi
        sum = sum + (mulfact * call[h]) // accumulate the weighted sum h
        = h - 1 // move to the next element
    }

    print sum // output the final result
}
```

Program:

```
#include<stdio.h>
#include<math.h>
```

```
int main(){
    int n;
    scanf("%d",&n);
    int cal[n];
    for(int i=0;i<n;i++){
        scanf("%d ",&cal[i]);
    }
    //sorting the array
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (cal[j] > cal[j+1]) {
                temp = cal[j];
                cal[j] = cal[j+1];
                cal[j+1] = temp;
            }
        }
    }
    int mulfact;
    int  sum=0;
    int h=n-1;
    for(int i=0;i<n;i++)
    {
        mulfact=pow(n,i);
        sum+=mulfact*cal[h];
        h--;
    }
    printf("%d",sum);
}
```

Output:



	Test	Input	Expected	Got	
✓	Test Case 1	3 1 3 2	18	18	✓
✓	Test Case 2	4 7 4 9 6	389	389	✓
✓	Test Case 3	3 5 10 7	76	76	✓