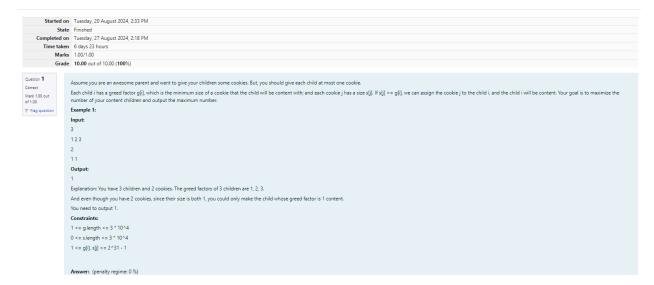
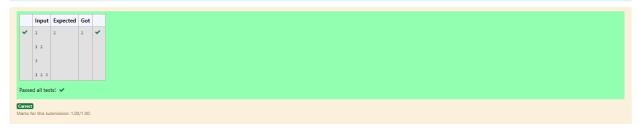
# Design and analysis of algorithm Greedy algorithm

October 1

Control 1





```
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<sup>i</sup> * 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<sup>0</sup> * 1) + (3<sup>1</sup> * 3) + (3<sup>2</sup> * 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
```

# For example:

Test	Input	Result
Test Case 1	3	18
	1 3 2	

Answer: (penalty regime: 0 %)

```
For example:
                     Input Result
  Test
  Test Case 1 3 18
 Answer: (penalty regime: 0 %)
Answer: (penalty regime: 0 %)

1  #include<stdio.h>
2  #include<math.h>
3  int main(){
4  int ajscanf("%
5  for(int i=0;1c
6  for(int i=0;1c
7  for(int i=0;1c
11  }
12  }
13  }
14  for(int i=0;1c
15  printf("%d", su
16 }
```

	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	~

# Passed all tests! 🗸

Correct
Marks for this submission: 1.00/1.00.

Question 1 Correct Mark 1.00 out of 1.00

```
Given an array of N integer, we have to maximize the sum of art[i] * i, where i is the index of the element (i = 0, 1, 2, ..., N). Write an algorithm based on Greedy technique with a Complexity O(nlogn).
```

First line specifies the number of elements-n

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

25340

Sample output: 40

Question 1
Correct
Mark 1.00 out
of 1.00
F Flag question

Given two arrays array\_One[] and array\_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs (1 element from each) is minimum. That is SUM (A[] \* B[I]) for all i is minimum.

## For example:

```
Input Result
     28
```

```
Answer: (penalty regime: 0 %)

1 #includes stdid.nh
2 #includes stdid.nh
3 *int compareAsc(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
    }
6 *int compareDesc(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
    }
9 *int main() {
    int array_One[n];
    int array_Two[n];
    int array_Two[n];
    if * for(int i = 0; i < n; i++) {
         scanf("Md", Sarray_Two[i]);
    }
    for(int i = 0; i < n; i++) {
         scanf("Md", Sarray_Two[i]);
    }
    gost(array_One, n, sizeof(int), compareAsc);
    acort(array_Two, n, rizeof(int), compareDesc);
    int sum = 0;
    for(int i = 0; i < n; i++) {
         scanf("Md", Sarray_Two[i]);
    }
    port(array_One, n, sizeof(int), compareDesc);
    int sum = 0;
    sort(array_Two, n, rizeof(int), compareDesc);
    int sum = 0;
    sort(int i = 0; i < n; i++) {
         sum + array_One[i] * array_Two[i];
    }
    printf("Md\n", sum);
    return 0;
}
                                                                                                                    | scanf("Md", Marray_Imot_;,,)
| gsort(array_One, n, sizeof(int), compareAsc);
| gsort(array_Two, n, sizeof(int), compareDesc);
| int sum = 0; i < n; i++) {
| sum += array_One[i] * array_Two[i];
| }
```

	Input	Expected	Got	
~	3	28	28	~
	1			
	2			
	3			
	4			
	5			
	6			
~	4	22	22	~