A Very Big Sum ☆

Calculate and print the sum of the elements in an array, keeping in mind that some of those integers may be quite large.

Input Format

The first line of the input consists of an integer n.

The next line contains n space-separated integers contained in the array.

Output Format

Print the integer sum of the elements in the array.

Constraints

$1 \leq n \leq 10$

$0 \leq ar[i] \leq 10^{10}$

Sample Input

```
5
1000000001 1000000002 1000000003 1000000004 1000000005
```

Output

```
5000000015
```

Note:

The range of the 32-bit integer is (-2^{31}) *to* $(2^{31} - 1)$ *or* $[-2147483648, 2147483647]$.

When we add several integer values, the resulting sum might exceed the above range. You might need to use long long int in C/C++ or long data type in Java to store such sums.

Author vatsalchanana
Difficulty Easy
Max Score 10
Submitted By 576368

NEED HELP?

View discussions
View editorial
View top submissions

RATE THIS CHALLENGE

☆ ☆ ☆ ☆ ☆

MORE DETAILS

Download problem statement
Download sample test cases
Suggest Edits

Current Buffer (saved locally, editable) C++

```cpp
#include <bits/stdc++.h>

using namespace std;

vector<string> split_string(string);

// Complete the aVeryBigSum function below.
long aVeryBigSum(vector<long> ar) {


}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    int ar_count;
    cin >> ar_count;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
```

```

    string ar_temp_temp;
    getline(cin, ar_temp_temp);

    vector<string> ar_temp = split_string(ar_temp_temp);

    vector<long> ar(ar_count);

    for (int i = 0; i < ar_count; i++) {
        long ar_item = stol(ar_temp[i]);

        ar[i] = ar_item;
    }

    long result = aVeryBigSum(ar);

    fout << result << "\n";

    fout.close();

    return 0;
}

vector<string> split_string(string input_string) {
    string::iterator new_end = unique(input_string.begin(), input_string.end(), [] (const char &x, const char &y) {
        return x == y and x == ' ';
    });

    input_string.erase(new_end, input_string.end());

    while (input_string[input_string.length() - 1] == ' ') {
        input_string.pop_back();
    }

    vector<string> splits;
    char delimiter = ' ';

    size_t i = 0;
    size_t pos = input_string.find(delimiter);

    while (pos != string::npos) {
        splits.push_back(input_string.substr(i, pos - i));

        i = pos + 1;
        pos = input_string.find(delimiter, i);
    }

    splits.push_back(input_string.substr(i, min(pos, input_string.length()) - i + 1));

    return splits;
}

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code