

Experiment-10

Student Name: Anjali Singh

Branch: CSE

Semester: 5th

Subject Name: Competitive Coding

UID: 20BCS9239

Section/Group: WM_607 A

Subject Code: 20CSP-314

PROBLEM STATEMENT 10.1

Marc loves cupcakes, but he also likes to stay fit. Each cupcake has a calorie count, and Marc can walk a distance to expend those calories. If Marc has eaten n cupcakes so far, after eating a cupcake with c calories he must walk *at least* c miles to maintain his weight.

Example

If he eats the cupcakes in the order shown, the miles he will need to walk are $[1, 3, 4]$. This is not the minimum, though, so we need to test other orders of consumption. In this case, our minimum miles is calculated as 6 .

Given the individual calorie counts for each of the cupcakes, determine the minimum number of miles Marc must walk to maintain his weight. Note that he can eat the cupcakes *in any order*.

Function Description

Complete the `marcsCakewalk` function in the editor below. `marcsCakewalk` has the following parameter(s):

- `int calorie[n]`: the calorie counts for each cupcake **Returns**
- `long`: the minimum miles necessary

Input Format

The first line contains an integer , the number of cupcakes in .

The second line contains space-separated integers, .

Constraints

□

□

Sample Input 0

3
1 3 2

Sample Output 0

11

Explanation 0

Let's say the number of miles Marc must walk to maintain his weight is . He can minimize by eating the cupcakes in the following order:

1. Eat the cupcake with calories, so .
2. Eat the cupcake with calories, so .
3. Eat the cupcake with calories, so .

We then print the final value of , which is , as our answer.

Sample Input 1

4
7 4 9 6

Sample Output 1

79

Solution:

```
#include <bits/stdc++.h> using namespace std;
typedef signed long long ll;
```

```
#undef _P #define _P(...) (void)printf(__VA_ARGS__) #define FOR(x,to) for(x=0;x<(to);x++) #define
FORR(x,arr) for(auto& x:arr) #define ITR(x,c) for(__typeof(c.begin()) x=c.begin();x!=c.end();x++) #define
ALL(a) (a.begin()),(a.end()) #define ZERO(a) memset(a,0,sizeof(a)) #define MINUS(a)
memset(a,0xff,sizeof(a))
```

```
int N; ll
C[1010];
```

```
void solve() { int i,j,k,l,r,x,y; string s;
```

```
cin>>N; FOR(i,N) cin>>C[i]; sort(C,C+N); reverse(C,C+N);
```

```
ll ret=0; FOR(i,N) ret +=
C[i]<<i; cout<<ret<<endl;
}
```

```
int main(int argc,char** argv){ string s;int i;
```

```
if(argc==1) ios::sync_with_stdio(false), cin.tie(0); FOR(i,argc-1) s+=argv[i+1],s+='\n'; FOR(i,s.size())  
ungetc(s[s.size()-1-i],stdin); solve(); return 0;  
}
```

OUTPUT:

Congratulations

You solved this challenge. Would you like to challenge your friends?



[Next Challenge](#)

✓ Test case 0

Compiler Message

Success

✓ Test case 1

✓ Test case 2

Input (stdin)

[Download](#)

```
1 3  
2 1 3 2
```

✓ Test case 3

✓ Test case 4

Expected Output

[Download](#)

✓ Test case 5

```
1 11
```

PROBLEM STATEMENT 10.2

Given a square grid of characters in the range `ascii[a-z]`, rearrange elements of each row alphabetically, ascending.

Determine if the columns are also in ascending alphabetical order, top to bottom. Return YES if they are or NO if they are not.

Example

The grid is illustrated below.

```
a b c a  
d e e f g
```

The rows are already in alphabetical order. The columns `a a e`, `b d f` and `c e g` are also in alphabetical order, so the answer would be YES. Only elements within the same row can be rearranged. They cannot be moved to a different row.

Function Description

Complete the *gridChallenge* function in the editor below.

gridChallenge has the following parameter(s):

- *string grid[n]*: an array of strings

Returns

- *string*: either YES or NO

Input Format

The first line contains , the number of testcases.

Each of the next sets of lines are described as follows:

- The first line contains , the number of rows and columns in the grid.
- The next lines contains a string of length

Constraints

Each string consists of lowercase letters in the range `ascii[a-z]`

Output Format

For each test case, on a separate line print YES if it is possible to rearrange the grid alphabetically ascending in both its rows and columns, or NO otherwise.

Sample Input

STDIN Function

1 t = 1 5 n = 5 ebacd grid = ['ebacd', 'fghij',

'olmkn', 'trpqs', 'xywuv'] fghij

olmkn trpqs xywuv

Sample Output

YES

Explanation

The x grid in the test case can be reordered to

abcde

fghij

klmno pqrst uvwxy

This fulfills the condition since the rows 1, 2, ..., 5 and the columns 1, 2, ..., 5 are all alphabetically sorted.

Solution:

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

using namespace std;

string ltrim(const string &); string
rtrim(const string &);

string gridChallenge(vector<string> grid) {
    for(int
i=0;i<grid.size();i++){          sort
(grid[i].begin(), grid[i].end() );    }
    for(int i=0;i<grid.size();i++){
vector<char> v;          for(int j=0;j<grid.size();j++){
        v.push_back(grid[j][i]);
    }          if(!is_sorted(v.begin(),v.end()
))
return "NO";
    }          return
"YES";
}

int main() {
    ofstream fout(getenv("OUTPUT_PATH"));
    string t_temp;
    getline(cin, t_temp);
```

```
int t = stoi(ltrim(rtrim(t_temp)));

for (int t_itr = 0; t_itr < t; t_itr++) {
string n_temp;          getline(cin, n_temp);

    int n = stoi(ltrim(rtrim(n_temp)));

    vector<string> grid(n);

    for (int i = 0; i < n; i++) {
string grid_item;          getline(cin, grid_item);

        grid[i] = grid_item;
    }

    string result = gridChallenge(grid);

    fout << result << "\n";
}
fout.close();

return 0; }

string ltrim(const string &str) {    string
s(str);

    s.erase(
        s.begin(),    find_if(s.begin(), s.end(),
not1(ptr_fun<int, int>(isspace)))
    );
return s;
}

string rtrim(const string &str) {
string s(str);

    s.erase(        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),    s.end()
    );
}
```

```
    return s;  
}
```

OUTPUT:

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

Test case 0

Test case 1 

Test case 2 

Test case 3 

Test case 4 

Test case 5 

Test case 6 

Success

Input (stdin)

```
1 1
2 5
3 eabcd
4 fghij
5 olkmn
6 trpqs
7 xywuv
```

[Download](#)

Expected Output

```
1 YES
```

[Download](#)

PROBLEM STATEMENT-10.3

You are given two arrays, and , both containing integers.

A pair of indices is *beautiful* if the element of array is equal to the element of array . In other words, pair is *beautiful* if and only if . A set containing beautiful pairs is called a *beautiful set*.

A beautiful set is called *pairwise disjoint* if for every pair belonging to the set there is no repetition of either or values.

For instance, if and the beautiful set is not pairwise disjoint as there is a repetition of , that is .

Your task is to change **exactly** element in so that the size of the pairwise disjoint beautiful set is maximum.

Function Description

Complete the *beautifulPairs* function in the editor below. It should return an integer that represents the maximum number of pairwise disjoint beautiful pairs that can be formed.

beautifulPairs has the following parameters:

- *A*: an array of integers
- *B*: an array of integers

Input Format

The first line contains a single integer , the number of elements in and .

The second line contains space-separated integers .

The third line contains space-separated integers .

Constraints

□

□

Output Format

Determine and print the maximum possible number of pairwise disjoint beautiful pairs.

Note: You must first change element in , and your choice of element must be optimal.

Sample Input 0

```
4
1 2 3 4
1 2 3 3
```

Sample Output 0

```
4
```

Explanation 0

You are given n and k .

The beautiful set is S and maximum sized pairwise disjoint beautiful set is either S_1 or S_2 .

We can do better. We change the k element of array from k to $k+1$. Now new B array is: B_1 and the pairwise disjoint beautiful set is S_1 . So, the answer is 4.

Note that we could have also selected index 3 instead of index 2 but it would have yielded the same result. Any other choice of index is not optimal.

Sample Input 1

```
6
3 5 7 11 5 8
5 7 11 10 5 8
```

Sample Output 1

```
6
```

SOLUTION:

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

```
using namespace std;
```

```
string ltrim(const string &); string rtrim(const  
string &); vector<string> split(const string &);
```

```
int beautifulPairs(vector<int> a, vector<int> b) {      unordered_multiset<int>  
set;  
    for(int e: a)  
set.insert(e);      int res =  
0;    for(int e:  
b) {      if(set.find(e) != set.end())  
{          res++;  
set.erase(set.find(e));  
    }  
}  
    if(set.size()) return res + 1;      else  
return res - 1;  
} int main()  
{  
    ofstream fout(getenv("OUTPUT_PATH"));  
    string n_temp;    getline(cin, n_temp);  
    int n =  
stoi(ltrim(rtrim(n_temp))  
);  
  
    string A_temp_temp;  
    getline(cin, A_temp_temp);  
  
    vector<string> A_temp = split(rtrim(A_temp_temp));  
  
    vector<int> A(n);  
  
    for (int i = 0; i < n; i++) {  
int A_item = stoi(A_temp[i]);  
  
        A[i] = A_item;  
    }  
    string B_temp_temp;
```

```
getline(cin, B_temp_temp);

vector<string> B_temp = split(rtrim(B_temp_temp));

vector<int> B(n);

for (int i = 0; i < n; i++) {
int B_item = stoi(B_temp[i]);

    B[i] = B_item;
}
int result = beautifulPairs(A, B);

fout << result << "\n";

fout.close();
return 0;
}
string ltrim(const string &str) {
string s(str);

    s.erase(
        s.begin(), find_if(s.begin(), s.end(),
not1(ptr_fun<int, int>(isspace)))
    );
return s;
}
string rtrim(const string &str) {
string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(), s.end()
    );
return s; }
vector<string> split(const string &str) {    vector<string>
tokens;
```

```
string::size_type start = 0;
string::size_type end = 0;

while ((end = str.find(" ", start)) != string::npos) {
    tokens.push_back(str.substr(start, end - start));

    start = end + 1;
}
tokens.push_back(str.substr(start));
return tokens;
}
```

OUTPUT:

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

✓ **Test case 0**

✓ Test case 1 

✓ Test case 2 

✓ Test case 3 

✓ Test case 4 

✓ Test case 5 

✓ Test case 6 

Compiler Message

Success

Input (stdin)

[Download](#)

1	4
2	1 2 3 4
3	1 2 3 3

Expected Output

[Download](#)

1	4
---	---

PROBLEM STATEMENT- 10.4

Alice is a kindergarten teacher. She wants to give some candies to the children in her class. All the children sit in a line and each of them has a rating score according to his or her performance in the class. Alice wants to give at least 1 candy to each child. If two children sit next to each other, then the one with the higher rating must get more candies. Alice wants to minimize the total number of candies she must buy.

Example

She gives the students candy in the following minimal amounts: . She must buy a minimum of 10 candies.

Function Description

Complete the *candies* function in the editor below. *candies* has the following parameter(s):

- *int n*: the number of children in the class
- *int arr[n]*: the ratings of each student

Returns

- *int*: the minimum number of candies Alice must buy

Input Format

The first line contains an integer, *n*, the size of *arr*.

Each of the next *n* lines contains an integer indicating the rating of the student at position *i*. **Constraints**

□

□

Sample Input 0

3
1
2

2

Sample Output 0

4

Explanation 0

Here 1, 2, 2 is the rating. Note that when two children have equal rating, they are allowed to have different number of candies. Hence optimal distribution will be 1, 2, 1.

Sample Input 1

10

2

4

2

6

1

7

8

9

2

1

Sample Output 1

19

Explanation 1

Optimal distribution will be

Sample Input 2

8

2

4

3

5
2
6
4
5

Sample Output 2

12

Explanation 2

Optimal distribution will be .

SOLUTION:

```
#include <bits/stdc++.h>
using namespace std;
string ltrim(const string
&); string
rtrim(const string &);

long candies(int n, vector<int> arr) {
vector<int>candies(n,1);
vector<int>candies_rev(n,1);    for (int
i=0, j=n-1; i<n-1; i++, j--){
if(arr[i] < arr[i+1]){
candies[i+1] += candies[i];      }
if(arr[j] < arr[j-1]){
candies_rev[j-1] += candies_rev[j];    }
}
for(int i=0; i<n; i++){
if(candies[i]<candies_rev[i]){
candies[i] = candies_rev[i];      }
}    long count = 0;    for(int
```

```
i=0;    i<candies.size();    i++){
count+=candies[i];    }    return
count; } int main() {
    ofstream fout(getenv("OUTPUT_PATH"));
    string n_temp;
    getline(cin, n_temp);

    int n = stoi(ltrim(rtrim(n_temp)));    vector<int> arr(n);

    for (int i = 0; i < n; i++) {        string
arr_item_temp;
    getline(cin, arr_item_temp);

        int arr_item = stoi(ltrim(rtrim(arr_item_temp)));

        arr[i] = arr_item;
    }
    long result = candies(n, arr);

    fout << result << "\n";

    fout.close();

    return 0;
}
string ltrim(const string &str) {    string
s(str);

    s.erase(
        s.begin(),    find_if(s.begin(), s.end(),
not1(ptr_fun<int, int>(isspace)))
    );
    return s;
}
string rtrim(const string &str) {
    string s(str);
```

```
s.erase( find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,  
int>(isspace))).base(), s.end()  
); return  
s; }
```

OUTPUT:

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

✓ Test case 0

✓ Test case 1 

✓ Test case 2 

✓ Test case 3 

✓ Test case 4 

✓ Test case 5 

✓ Test case 6 

Compiler Message

Success

Input (stdin)

[Download](#)

1	3
2	1
3	2
4	2

Expected Output

[Download](#)

1	4
---	---