Write a program to take value V and  we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin.

Output Format:

print the integer which is change of the  number.

Example Input :

64

Output:

4

Explanaton:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

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

```c
1   #include<stdio.h>
2   int main(){
3       int v;
4       scanf("%d",&v);
5       int d[]={1000,500,100,50,20,10,5,2,1};
6       int n=sizeof(d)/sizeof(d[0]);
7       int c=0;
8       for(int i=0;i<n;i++){
9           while(v>=d[i])
10          {
11          v-=d[i];
12          c++;
13
14          }
15      }
16      printf("%d",c);
17
18
19  }
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 49 | 5 | 5 | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Jump to...

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

**Example 1:**

**Input:**

3

1 2 3

2

1 1

**Output:**

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

**Constraints:**

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1

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

```c
1  #include<stdio.h>
2  int main()
3  {
4      int x,y,count=0;
5      scanf("%d",&x);
6      int a[x];
7      for(int i=0;i<x;i++)
8      {
9          scanf("%d",&a[i]);
10     }
11     scanf("%d",&y);
12     int b[y];
13     for(int i=0;i<y;i++) {
14         scanf("%d",&b[i]);
15     }
16     for(int i=0;i<y;i++)
17     {
18         if(a[i]==b[i])
19         {
20             count++;
21         }
22     }
23     printf("%d",count);
24 }
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 2<br><br>1  2<br><br>3<br><br>1  2  3 | 2 | 2 | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Jump to...

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

**For example:**

| Test | Input | Result |
|------|-------|--------|
| Test Case 1 | 3<br>1 3 2 | 18 |

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

```c
1  #include<stdio.h>
2  #include<math.h>
3  void selectionSort(int arr[], int n)
4  {
5  for (int i = 0; i < n - 1; i++)
6  {
7  int min_idx = i;
8
9  for (int j = i + 1; j < n; j++)
10 {
11 if (arr[j] > arr[min_idx])
12 {
13     min_idx = j;
14 }
15 }
16 int temp = arr[min_idx];
17 arr[min_idx] = arr[i];
18 arr[i] = temp;
19 }
20 }
21 int main()
22 {
23 int n;
24 scanf("%d",&n);
25 int arr[n];
26 for(int i = 0; i< n; i++){
27     scanf("%d",&arr[i]);
28 }
29 selectionSort(arr,n);
30 int s = 0;
31 for(int i = 0; i< n; i++){
32     s+= pow(n,i) * arr[i];
33 }
```

```
34  printf("%d",s);
35  }
36
37
```

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✔ | Test Case 1 | 3<br>1 3 2 | 18 | 18 | ✔ |
| ✔ | Test Case 2 | 4<br>7 4 9 6 | 389 | 389 | ✔ |
| ✔ | Test Case 3 | 3<br>5 10 7 | 76 | 76 | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Jump to...

Given an array of N integer, we have to maximize the sum of arr[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).

 Input Format:

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:

5

2 5 3 4 0

Sample output:

40

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

```c
1  #include <stdio.h>
2  #include <stdlib.h>
3  int compare(const void *a, const void *b) {
4      return (*(int*)a - *(int*)b);
5  }
6  int main() {
7      int n;
8      scanf("%d", &n);
9      int arr[n];
10     for (int i = 0; i < n; i++) {
11         scanf("%d", &arr[i]);
12     }
13     qsort(arr, n, sizeof(int), compare);
14     int maxSum = 0;
15     for (int i = 0; i < n; i++) {
16         maxSum += arr[i] * i;
17     }
18     printf("%d\n", maxSum);
19 }
20
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 5<br>2<br>5<br>3<br>4<br>0 | 40 | 40 | ✔ |
| ✔ | 10<br>2<br>2<br>2<br>4<br>4<br>3<br>3<br>5<br>5<br>5 | 191 | 191 | ✔ |

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 2<br>45<br>3 | 45 | 45 | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Jump to...

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[i] * B[i]) for all i is minimum.

**For example:**

| Input | Result |
|-------|--------|
| 3<br>1<br>2<br>3<br>4<br>5<br>6 | 28 |

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

```c
#include <stdio.h>
void sortArray(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
int main() {
    int n;
    scanf("%d", &n);
    int array_One[n], array_Two[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_One[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_Two[i]);
    }
    sortArray(array_One, n);
    sortArray(array_Two, n);
    int start = 0;
    int end = n - 1;
    while (start < end) {
        int temp = array_Two[start];
        array_Two[start] = array_Two[end];
        array_Two[end] = temp;
        start++;
        end--;
    }
    int minSum = 0;
    for (int i = 0; i < n; i++) {
        minSum += array_One[i] * array_Two[i];
    }
    printf("%d\n", minSum);
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 3<br>1<br>2<br>3<br>4<br>5<br>6 | 28 | 28 | ✔ |
| ✔ | 4<br>7<br>5<br>1<br>2<br>1<br>3<br>4<br>1 | 22 | 22 | ✔ |
| ✔ | 5<br>20<br>10<br>30<br>10<br>40<br>8<br>9<br>4<br>3<br>10 | 590 | 590 | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Jump to...