

NAME: Lejo C

REG NO: 241801136

## GREEDY ALGORITHMS PROGRAM 1:

Question 1 | Correct | Mark 1.00 out of 1.00 | [Flag question](#)

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 %)

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

	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

2:

# PROGRAM

**Question 1** | Correct | Mark 1.00 out of 1.00 | [Flag question](#)

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  $s[j] \geq g[i]$ , 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 %)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int cmp(const void *a, const void *b)
5 {
6     int x = *(int*)a;
7     int y = *(int*)b;
8     return (x-y);
9 }
10
11 int main() {
12     int n, m;
13     scanf("%d", &n);
14     int g[n];
15     for (int i = 0; i < n; i++)
16     {
17         scanf("%d", &g[i]);
18     }
19     scanf("%d", &m);
20     int s[m];
21     for (int i = 0; i < m; i++)
22     {
23         scanf("%d", &s[i]);
24     }
25     qsort(g, n, sizeof(int), cmp);
26     qsort(s, m, sizeof(int), cmp);
27     int i = 0, j = 0;
28     int content_children = 0;
29     while (i < n && j < m)
30     {
31         if (s[j] >= g[i])
32         {
33             content_children++;
34             i++;
35             j++;
36         }
37         else
38         {
39             j++;
40         }
41     }
42     printf("%d\n", content_children);
43     return 0;
44 }
45
```

	Input	Expected	Got	
✓	2	2	2	✓
	1 2			
	3			
	1 2 3			

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

# PROGRAM

## 3:

Question 1 | Correct Mark 1.00 out of 1.00 [Flag question](#)

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	18
	1 3 2	

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

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<math.h>
4 int cmp_desc(const void *a, const void *b)
5 {
6     return (*(int*)b - *(int*)a);
7 }
8
9 int main() {
10     int n;
11     scanf("%d", &n);
12     int a[n];
13     for (int i = 0; i < n; i++) {
14         scanf("%d", &a[i]);
15     }
16     qsort(a, n, sizeof(int), cmp_desc);
17     int c=0;
18     for(int i=0;i<n;i++)
19     {
20         c+=(pow(n,i)*a[i]);
21     }
22     printf("%d",c);
23 }
```

	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.

**4:**

# PROGRAM

Question 1 | Correct | Mark 1.00 out of 1.00 | [Flag question](#)

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

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 %)

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int cmp(const void *a, const void *b)
4 {
5     return (*(int*)a - *(int*)b);
6 }
7
8 int main()
9 {
10     int n;
11     scanf("%d",&n);
12     int a[n],sum=0;
13     for(int i=0;i<n;i++)
14     {
15         scanf("%d",&a[i]);
16     }
17     qsort(a,n,sizeof(int),cmp);
18     for(int i=0;i<n;i++)
19     {
20         sum+=a[i]*i;
21     }
22     printf("%d",sum);
23 }
24
```

	Input	Expected	Got	
✓	5	40	40	✓
	2			
	5			
	3			
	4			
	0			
✓	10	191	191	✓
	2			
	2			
	2			
	4			
	4			
	3			
	3			
	5			
	5			
	5			
✓	2	45	45	✓
	45			
	3			

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

5:



# PROGRAM

Question 1 | Correct | Mark 1.00 out of 1.00 | [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  $\text{SUM}(A[i] * B[i])$  for all `i` is minimum.

For example:

Input	Result
3	28
1	
2	
3	
4	
5	
6	

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

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int asc(const void *a, const void *b)
4 {
5     return (*(int*)a - *(int*)b);
6 }
7 int desc(const void *a, const void *b) {
8     return (*(int*)b - *(int*)a);
9 }
10 int main()
11 {
12     int n;
13     scanf("%d",&n);
14     int a[n],b[n],sum=0;
15     for(int i=0;i<n;i++)
16     {
17         scanf("%d",&a[i]);
18     }
19     for(int i=0;i<n;i++)
20     {
21         scanf("%d",&b[i]);
22     }
23     qsort(a,n,sizeof(int),asc);
24     qsort(b,n,sizeof(int),desc);
25     for(int i=0;i<n;i++)
26     {
27         sum+=a[i]*b[i];
28     }
29     printf("%d",sum);
30 }
```



	Input	Expected	Got	
✓	3	28	28	✓
	1			
	2			
	3			
	4			
	5			
	6			
✓	4	22	22	✓
	7			
	5			
	1			
	2			
	1			
	3			
	4			
	1			
✓	5	590	590	✓
	20			
	10			
	30			
	10			
	40			
	8			
	9			
	4			
	3			
	10			