Started on	Tuesday, 8 October 2024, 1:45 PM
State	Finished
Completed on	Tuesday, 8 October 2024, 1:47 PM
Time taken	2 mins 49 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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
#include <stdio.h>

int min_coins(int v) {
    int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int count = 0;
    int size = sizeof(denominations) / sizeof(denominations[0]);

    for (int i = 0; i < size; i++) {
        if (v == 0) {
            break;
        }
        count += v / denominations[i];
        v = v % denominations[i];
    }

    return count;
}

int main() {
    int value;
    scanf("%d",&value);
    int result = min_coins(value);
    printf("%d",result);
    return 0;
}

```

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 6-Implementation of Quick Sort

Jump to...

2-G-Cookies Problem ►

Started on	Tuesday, 8 October 2024, 1:48 PM
State	Finished
Completed on	Tuesday, 8 October 2024, 1:50 PM
Time taken	1 min 59 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

int max_content_children(int *greed_factors, int g_length, int *cookie_sizes, int s_length) {
    qsort(greed_factors, g_length, sizeof(int), compare);
    qsort(cookie_sizes, s_length, sizeof(int), compare);

    int child_i = 0;
    int cookie_j = 0;
    int content_children = 0;

    while (child_i < g_length && cookie_j < s_length) {
        if (cookie_sizes[cookie_j] >= greed_factors[child_i]) {
            content_children++;
            child_i++;
        }
        cookie_j++;
    }

    return content_children;
}

int main() {
    int n, m;
    scanf("%d", &n);
    int *greed_factors = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
```

```
32          scanf("%d", &greed_factors[i]);
33      }
34      scanf("%d", &m);
35      int *cookie_sizes = (int *)malloc(m * sizeof(int));
36      for (int i = 0; i < m; i++) {
37          scanf("%d", &cookie_sizes[i]);
38      }
39
40      int result = max_content_children(greed_factors, n, cookie_sizes, m);
41      printf("%d\n", result);
42
43      free(greed_factors);
44      free(cookie_sizes);
45      return 0;
46  }
47
```

	Input	Expected	Got	
✔	2 1 2 3 1 2 3	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 1-G-Coin Problem

Jump to...

3-G-Burger Problem ►

Started on	Tuesday, 8 October 2024, 1:50 PM
State	Finished
Completed on	Tuesday, 8 October 2024, 1:51 PM
Time taken	1 min 1 sec
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

long long maximize_sum(int *arr, int n) {
    qsort(arr, n, sizeof(int), compare);
    long long max_sum = 0;

    for (int i = 0; i < n; i++) {
        max_sum += (long long)arr[i] * i;
    }

    return max_sum;
}

int main() {
    int n;
    scanf("%d", &n);
    int *arr = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    long long result = maximize_sum(arr, n);
    printf("%lld\n", result);

    free(arr);
    return 0;
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 3-G-Burger Problem

Jump to...

5-G-Product of Array elements-Minimum ►

Started on	Tuesday, 8 October 2024, 1:54 PM
State	Finished
Completed on	Tuesday, 8 October 2024, 1:56 PM
Time taken	1 min 19 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 1 2 3 4 5 6	28

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>

int compare_asc(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

int compare_desc(const void *a, const void *b) {
    return (*(int *)b - *(int *)a);
}

long long minimum_product_sum(int *array_one, int *array_two, int n) {
    qsort(array_one, n, sizeof(int), compare_asc);
    qsort(array_two, n, sizeof(int), compare_desc);

    long long min_sum = 0;
    for (int i = 0; i < n; i++) {
        min_sum += (long long)array_one[i] * array_two[i];
    }

    return min_sum;
}

int main() {
    int n;
    scanf("%d", &n);
    int *array_one = (int *)malloc(n * sizeof(int));
    int *array_two = (int *)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_one[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_two[i]);
    }

    long long result = minimum_product_sum(array_one, array_two, n);
    printf("%lld\n", result);

    free(array_one);
    free(array_two);
    return 0;
}

```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 4-G-Array Sum max problem

Jump to...

1-DP-Playing with Numbers ►