Started on	Tuesday, 10 September 2024, 1:38 PM
State	Finished
Completed on	Tuesday, 10 September 2024, 1:54 PM
Time taken	16 mins 26 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 main() {
    int value;

    scanf("%d", &value);
    int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int numDenominations = sizeof(denominations) / sizeof(denominations[0]);

    int count = 0;

    for (int i = 0; i < numDenominations; i++) {

        if (value >= denominations[i]) {
            count += value / denominations[i];
            value %= denominations[i];
        }
    }


    printf("%d\n", count);

    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, 10 September 2024, 1:58 PM
State	Finished
Completed on	Tuesday, 10 September 2024, 2:04 PM
Time taken	5 mins 51 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 main() {
    int numChildren, numCookies;


    scanf("%d", &numChildren);
    int *greedFactors = (int*)malloc(numChildren * sizeof(int));


    for (int i = 0; i < numChildren; i++) {
        scanf("%d", &greedFactors[i]);
    }


    scanf("%d", &numCookies);
    int *cookieSizes = (int*)malloc(numCookies * sizeof(int));


    for (int j = 0; j < numCookies; j++) {
        scanf("%d", &cookieSizes[j]);
    }


    qsort(greedFactors, numChildren, sizeof(int), compare);
    qsort(cookieSizes, numCookies, sizeof(int), compare);
```

```
33
34      int childIndex = 0;
35      int cookieIndex = 0;
36      int contentChildren = 0;
37
38      while (childIndex < numChildren && cookieIndex < numCookies) {
39
40          if (cookieSizes[cookieIndex] >= greedFactors[childIndex]) {
41
42              contentChildren++;
43              childIndex++;
44          }
45          cookieIndex++;
46      }
47
48      printf("%d\n", contentChildren);
49
50
51      free(greedFactors);
52      free(cookieSizes);
```

	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, 10 September 2024, 2:07 PM
State	Never submitted

Question **1**

Incorrect

Mark 0.00 out of 1.00

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 1 3 2	18

Answer: (penalty regime: 0 %)

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

// Comparison function to sort in descending order
int compare(const void *a, const void *b) {
    return (*(int *)b - *(int *)a);
}

int main() {
    int n;

    // Input: number of burgers
    scanf("%d", &n);

    int calories[n];

    // Input: calories of each burger
    for (int i = 0; i < n; i++) {
        scanf("%d", &calories[i]);
    }

    // Sort the array of calories in descending order
    qsort(calories, n, sizeof(int), compare);

    long long total_distance = 0;

```

```
    // Calculate the minimum distance to burn the calories
    for (int i = 0; i < n; i++) {
        // Calculate 3^i and multiply with calories[i]
        long long multiplier = 1;
        for (int j = 0; j < i; j++) {
            multiplier *= 3; // Calculate the power of 3 incrementally
        }
        total_distance += multiplier * calories[i]; // Add to total distance
    }

    // Output: Minimum distance to burn the calories
    printf("%lld\n", total_distance);

    return 0;
}
```

Check

	Test	Input	Expected	Got	
✔	Test Case 1	3 1 3 2	18	18	✔
✔	Test Case 3	3 5 10 7	76	76	✔

Your code failed one or more hidden tests.

Your code must pass all tests to earn any marks. Try again.

Incorrect

Marks for this submission: 0.00/1.00.

◄ 2-G-Cookies Problem

Jump to...

4-G-Array Sum max problem ►

Started on	Tuesday, 10 September 2024, 2:41 PM
State	Finished
Completed on	Monday, 30 September 2024, 9:08 AM
Time taken	19 days 18 hours
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>

// Comparator function for qsort to sort in ascending order
int compareAscending(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

int main() {
    int n;
    // Read the number of elements
    scanf("%d", &n);

    // Allocate memory for the array
    int *arr = (int *)malloc(n * sizeof(int));

    // Read the array elements
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Sort the array in ascending order
    qsort(arr, n, sizeof(int), compareAscending);

    // Calculate the maximum sum
    long long maxSum = 0;
    for (int i = 0; i < n; i++) {
        maxSum += (long long)arr[i] * i;
    }

    // Output the result
    printf("%lld\n", maxSum);

    // Free the allocated memory
    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, 10 September 2024, 2:45 PM
State	Finished
Completed on	Tuesday, 10 September 2024, 2:46 PM
Time taken	56 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>

// Comparator function for qsort to sort in ascending order
int compareAscending(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

// Comparator function for qsort to sort in descending order
int compareDescending(const void *a, const void *b) {
    return (*(int *)b - *(int *)a);
}

int main() {
    int n;
    // Read the number of elements
    scanf("%d", &n);

    // Allocate memory for the arrays
    int *array_One = (int *)malloc(n * sizeof(int));
    int *array_Two = (int *)malloc(n * sizeof(int));

    // Read the elements of array_One
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_One[i]);
    }

    // Read the elements of array_Two
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_Two[i]);
    }

    // Sort array_One in ascending order
    qsort(array_One, n, sizeof(int), compareAscending);

    // Sort array_Two in descending order
    qsort(array_Two, n, sizeof(int), compareDescending);

    // Calculate the minimum sum of products
    long long minSum = 0;
    for (int i = 0; i < n; i++) {
        minSum += (long long)array_One[i] * array_Two[i];
    }

    // Output the result
    printf("%lld\n", minSum);

    // Free the allocated memory
```

```
49      free(array_One);
50      free(array_Two);
51
52      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 ►