**GREEDY ALGORITHMS:**

**AIM:**

1-G coin problem.

**ALGORITHM:**

1. Read integer `v` (the amount of money).

2. Initialize counter `c` to 0.

3. Define array `d` with denominations: `{1000, 500, 100, 50, 20, 10, 5, 2, 1}`.

4. Loop through each denomination in `d`:

- For each index `i` from `0` to `8`:

- Update counter `c` by adding the quotient of `v` divided by `d[i]`.

- Update `v` to the remainder of `v` divided by `d[i]`.

5. Print the total count of coins/notes (`c`).

**PROBLEM:**

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

Explanation:

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

**PROGRAM:**

#include <stdio.h>

int main()

{

int v,c=0;

int d[]= {1000, 500, 100, 50, 20, 10, 5, 2, 1};

scanf("%d",&v);

for(int i=0;i<9;i++)

{

c+=v/d[i];

v%=d[i];

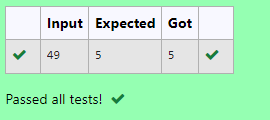
}

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

return 0;

}

**OUTPUT:**



**AIM:**

2-G cookies problem.

**ALGORITHM:**

1. Read integer `m` (size of the first array).

2. Create array `g` of size `m`.

3. Fill `g` with user input.

4. Read integer `n` (size of the second array).

5. Create array `s` of size `n`.

6. Fill `s` with user input.

7. Initialize counter `count` to 0.

8. Loop through each index `i` from `0` to `m-1`:

- For each index `j` from `0` to `n-1`:

- If `g[i] <= s[j]`:

- Increment `count`.

- Increment index `i`.

9. Print the value of `count - 1`.

**PROBLEM:**

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

**PROGRAM:**

#include <stdio.h>

int main(){

int m,n,count=0;

scanf("%d",&m);

int g[m];

for (int i=0;i<m;i++)

{

scanf("%d",&g[i]);

}

scanf("%d",&n);

int s[n];

for (int i=0;i<n;i++)

{

scanf("%d",&s[i]);

}

for (int i=0;i<m;i++)

{

for (int j=0;j<n;j++)

{

if (g[i]<=s[i])

{

count++;

i++;

}

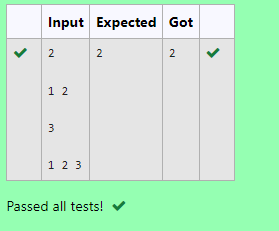
}

}

printf("%d",count-1);

}

**OUTPUT:**



**AIM:**

3-G burger problem.

**ALGORITHM:**

1. Read integer `n` (size of the array).

2. Create array `c` of size `n`.

3. Fill `c` with user input.

4. Sort the array `c` in descending order using a nested loop:

- For each index `i` from `0` to `n-1`:

- For each index `j` from `i + 1` to `n-1`:

- If `c[i] < c[j]`, swap `c[i]` and `c[j]`.

5. Initialize variable `d` to 0.

6. Calculate the value of `d` using the formula:

- For each index `i` from `0` to `n-1`:

- Update `d` by adding `pow(n, i) \* c[i]`.

7. Print the final value of `d`.

**PROBLEM:**

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 *3i \* 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 (30 \* 1) + (31 \* 3) + (32 \* 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 |

**PROGRAM:**

#include <stdio.h>

#include <math.h>

int main()

{

int n;

scanf("%d",&n);

int c[n];

for (int i=0;i<n;i++)

{

scanf("%d", &c[i]);

}

for (int i=0;i<n;i++)

{

for (int j=i+1;j<n;j++)

{

if (c[i]<c[j])

{

int temp=c[i];

c[i]=c[j];

c[j]=temp;

}

}

}

int d=0;

for (int i=0;i<n;i++)

{

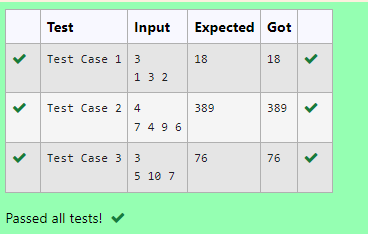
d+=pow(n,i)\*c[i];

}

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

}

**OUTPUT:**

****

**AIM:**

4-G Array sum max problem.

**ALGORITHM:**

1. Read integer `n` (size of the array).

2. Create array `arr` of size `n`.

3. Fill `arr` with user input.

4. Sort the array `arr` in descending order using a nested loop:

- For each index `i` from `0` to `n-1`:

- For each index `j` from `i + 1` to `n-1`:

- If `arr[i] < arr[j]`, swap the elements `arr[i]` and `arr[j]`.

5. Initialize variable `sum` to 0.

6. Calculate the sum using the formula:

- For each index `i` from `0` to `n-1`:

- Update `sum` by adding the product of `arr[i]` and `(n - i - 1)`.

7. Print the final value of `sum`.

**PROBLEM:**

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

**PROGRAM:**

#include <stdio.h>

int main()

{

int n;

scanf("%d",&n);

int arr[n];

for(int i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

for(int i=0;i<n;i++)

{

for(int j=i+1;j<n;j++)

{

if(arr[i]<arr[j])

{

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

int sum=0;

for(int i=0;i<n;i++)

{

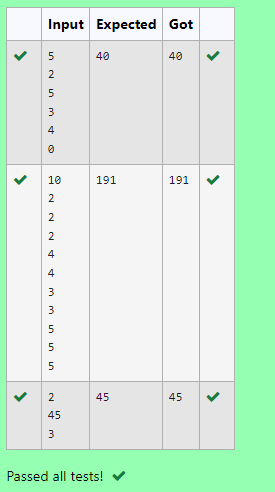
sum+=arr[i]\*(n-i-1);

}

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

}

**OUTPUT:**



**AIM:**

5-G Product of Array elements - Minimum

**ALGORITHM:**

1. Read integer `n` (size of the arrays).

2. Create two arrays, `arr1` and `arr2`, both of size `n`.

3. Fill `arr1` with user input.

4. Fill `arr2` with user input.

5. Sort `arr1` in ascending order using a nested loop:

- For each index `i` from `0` to `n-1`:

- For each index `j` from `i + 1` to `n-1`:

- If `arr1[i] > arr1[j]`, swap the elements `arr1[i]` and `arr1[j]`.

6. Sort `arr2` in descending order using a nested loop:

- For each index `i` from `0` to `n-1`:

- For each index `j` from `i + 1` to `n-1`:

- If `arr2[i] < arr2[j]`, swap the elements `arr2[i]` and `arr2[j]`.

7. Initialize variable `sum` to 0.

8. Calculate the sum of products of corresponding elements from the sorted arrays:

- For each index `i` from `0` to `n-1`:

- Update `sum` by adding the product of `arr1[i] \* arr2[i]`.

9. Print the final value of `sum`.

**PROBLEM:**

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 |

**PROGRAM:**

#include <stdio.h>

int main()

{

int n;

scanf("%d",&n);

int arr1[n],arr2[n];

for (int i=0;i<n;i++)

{

scanf("%d",&arr1[i]);

}

for (int i=0;i<n;i++)

{

scanf("%d",&arr2[i]);

}

for (int i=0;i<n;i++)

{

for (int j=i+1;j<n;j++)

{

if (arr1[i]>arr1[j])

{

int temp=arr1[i];

arr1[i]=arr1[j];

arr1[j]=temp;

}

}

}

for (int i=0;i<n;i++)

{

for (int j=i+1;j<n;j++)

{

if (arr2[i]<arr2[j])

{

int temp=arr2[i];

arr2[i]=arr2[j];

arr2[j]=temp;

}

}

}

int sum=0;

for (int i=0;i<n;i++)

{

sum+=arr1[i]\*arr2[i];

}

printf("%d",sum);

}

**OUTPUT:**

