**03 - Greedy Algorithms**

|  |  |
| --- | --- |
| **Ex. No. : 3.1** | **Date: 26.08.24** |
| **Register No.: 230701363** | **Name : D.Theepak Palani Kumar** |



AIM:

**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.**

ALGORITHM:

Step 1: Start

Step 2: Initialize an array currency with denominations and read the value of n from the user.

Step 3: Initialize a variable count to 0 and a variable j to 0.

Step 4: Use a while loop to find the first denomination in currency that is less than or equal to n.

Step 5: While n is not 0, check if the current currency[j] is less than or equal to n. If true, update count by adding the integer division of n by currency[j], then update n using the remainder of that division. Increment j.

Step 6: Print the value of count, which represents the total number of currency notes/coins used.

Step 7: End

PROGRAM:

#include<stdio.h> int main()

{

int currency[]={1000,500,100,50,20,10,5,2,1}; int n,count=0; scanf("%d",&n); int j = 0; while(currency[j]>n){ j++;

}

while(n!=0){ if(currency[j]<n){ count+=n/currency[j]; n=n%currency[j];

}

j++;

}

printf("%d",count);

}

OUTPUT:



RESULT:

Hence the above program has been executed successfully.

|  |  |
| --- | --- |
| **Ex. No. : 3.2** | **Date: 26.08.24** |
| **Register No.: 230701363** | **Name : D.Theepak Palani Kumar** |



AIM:

**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.**

ALGORITHM:

Step 1: Start

Step 2: Read the value of c (the number of groups) from the user. Initialize an array g of size c and read c values into it.

Step 3: Read the value of n (the number of elements) from the user. Initialize an array s of size n and read n values into it.

Step 4: Initialize a variable count to 0.

Step 5: Use a nested loop to compare each element in g with elements in s. If any s[j] is greater than or equal to g[i], increment count and break the inner loop.

Step 6: Print the value of count, representing the number of successful comparisons.

Step 7: End

PROGRAM:

#include<stdio.h> int main(){ int c,n,count=0; scanf("%d",&c); int g[c]; for(int i=0;i<c;i++) { scanf("%d",&g[i]);} int s[n]; scanf("%d",&n); for(int i=0;i<n;i++){ scanf("%d",&s[i]);} for(int i=0;i<c;i++){ for(int j=0;j<n;j++){ if(s[j]>=g[i]){

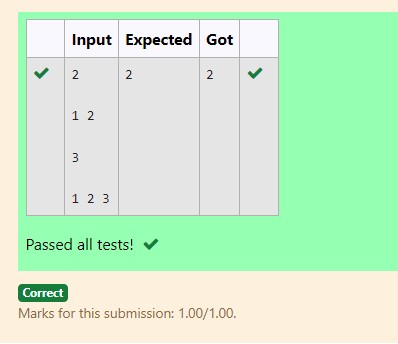
count++; break;

}}}

printf("%d",count);

}

OUTPUT:



RESULT:

Hence the above program has been executed successfully.

|  |  |
| --- | --- |
| **Ex. No. : 3.3** | **Date: 26.08.24** |
| **Register No.: 230701363** | **Name : D.Theepak Palani Kumar** |



AIM:

**A person needs to eat burgers. Each burger contains a count of calories. 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 burgers 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-separated 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**

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array cal of size n. Read n values into the array.

Step 3: Use a nested loop to sort the array cal in descending order using the bubble sort algorithm.

Step 4: Initialize a variable s to 0.

Step 5: Loop through the sorted array and calculate the value of s by summing pow(n, i) \* cal[i] for each index i.

Step 6: Print the value of s. Step 7: End

PROGRAM:

#include<stdio.h> #include<math.h> int main()

{

int n,km=0; scanf("%d",&n); int cal[n]; for(int i=0;i<n;i++)

{

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

}

for(int i=0;i<n-1;i++)

{

for(int j=0;j<n-i-1;j++)

{ if(cal[j]<cal[j+1])

{

int temp=cal[j]; cal[j]=cal[j+1]; cal[j+1]=temp;

}

}

}

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

{

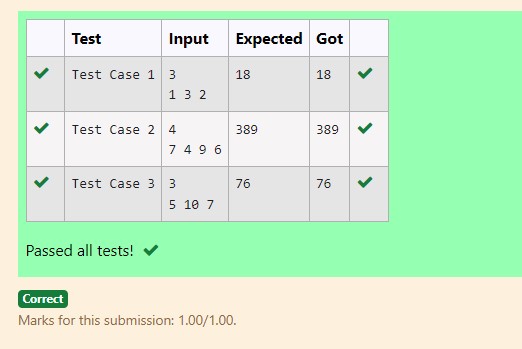
km+=pow(n,i)\*cal[i];

}

printf("%d",km);

}

OUTPUT:



RESULT:

Hence the above program has been executed successfully.

|  |  |
| --- | --- |
| **Ex. No. : 3.4** | **Date: 26.08.24** |
| **Register No.: 230701363** | **Name : D.Theepak Palani Kumar** |



AIM:

**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**

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize an array arr of size n. Read n values into the array.

Step 3: Use the qsort function to sort the array arr in ascending order by calling the compare function.

Step 4: Initialize a variable s to 0.

Step 5: Loop through the sorted array and calculate the value of s by summing arr[i] \* i for each index i.

Step 6: Print the value of s.

Step 7: End

PROGRAM:

#include<stdio.h> int main()

{

int n,sum=0; scanf("%d",&n); int arr[n]; for(int i=0;i<n;i++){ scanf("%d",&arr[i]);

}for(int i=1;i<n;i++){ int j=i; int temp=arr[j]; while(j>0 && arr[j-1]>temp){

arr[j]=arr[j-1];

j--;

}

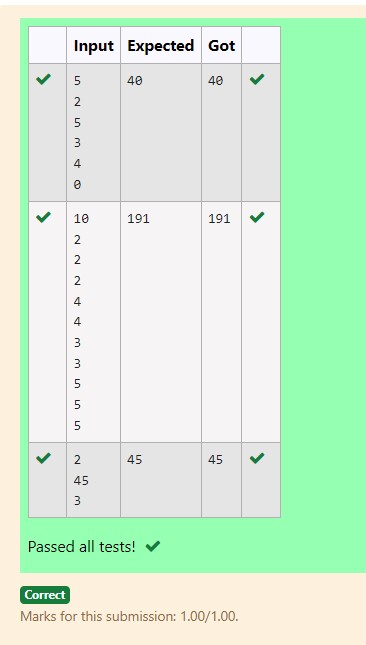
arr[j]=temp;} for(int i=0;i<n;i++){ sum+=arr[i]\*i;

}

printf("%d",sum);

}

OUTPUT:



RESULT:

Hence the above program has been executed successfully.

|  |  |
| --- | --- |
| **Ex. No. : 3.5** | **Date: 26.08.24** |
| **Register No.: 230701363** | **Name : D.Theepak Palani Kumar** |



AIM:

**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** |

ALGORITHM:

Step 1: Start

Step 2: Read the value of n from the user and initialize two arrays, arr1 and arr2, each of size n. Read n values into both arrays.

Step 3: Use the qsort function to sort arr1 in ascending order using compare1 and arr2 in descending order using compare2.

Step 4: Initialize a variable s to 0.

Step 5: Loop through both sorted arrays and calculate the value of s by summing arr1[i] \* arr2[i] for each index i.

Step 6: Print the value of s.

Step 7: End

PROGRAM:

#include<stdio.h> void sort(int a[],int x)

{

for(int i=1;i<x;i++)

{

int j=i; int temp=a[j]; while(j>0 && a[j-1]>temp)

{ a[j]=a[j-1];

j--;

}

a[j]=temp;

}

}

int main()

{

int n,sum=0;

scanf("%d",&n); int a[n],b[n]; for(int i=0;i<n;i++)

{

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

}

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

{

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

} sort(a,n); sort(b,n); for(int i=0;i<n;i++)

{

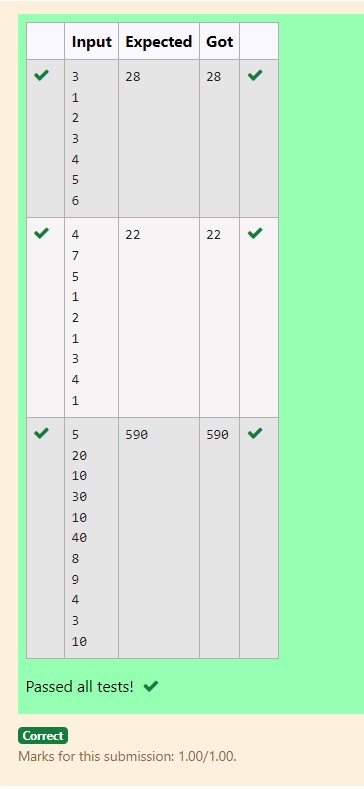
sum+=a[i]\*b[n-i-1];

}

printf("%d",sum);

}

OUTPUT:



RESULT:

Hence the above program has been executed successfully.