DESIGN AND ANALYSIS OF ALGORITHMS

-ARITRA GUPTA

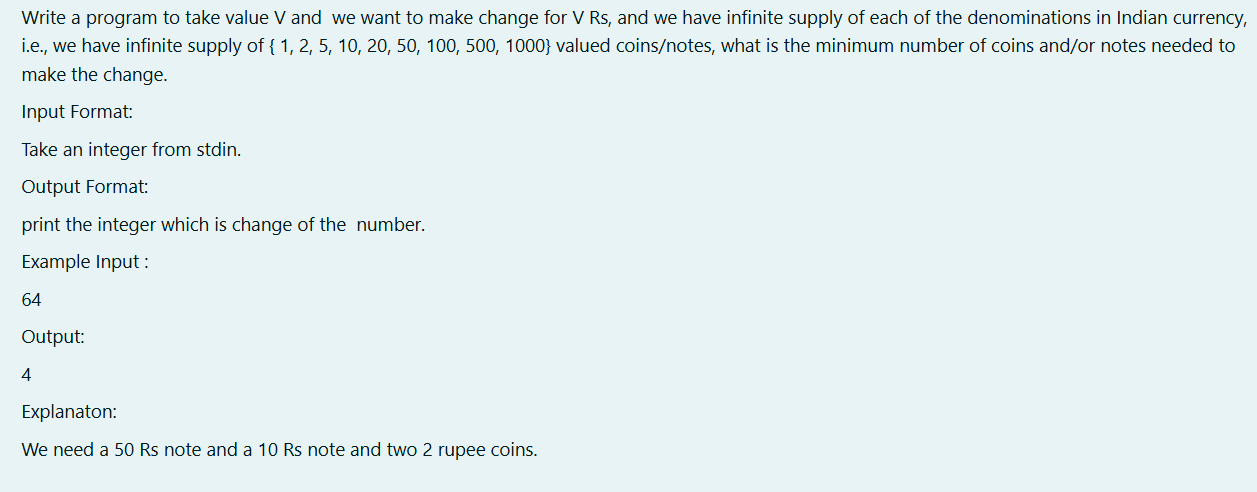
230701033

CSE –‘A’

GREEDY ALGORITHMS

EX 3.1

AIM:



SOURCE CODE:

#include<stdio.h>

int main(){

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

int c=0,val,sum=0;

scanf("%d",&val);

do{

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

if(sum+arr[i]<=val){

sum+=arr[i];

c++;

break;

}

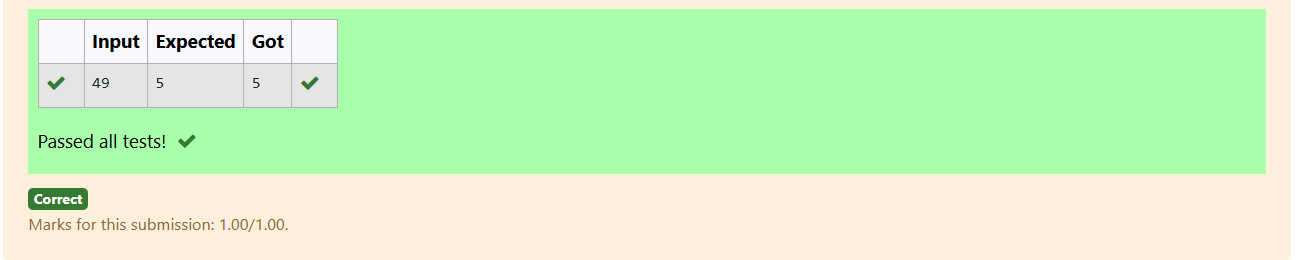
}

}while(sum!=val);

printf("%d",c);

}

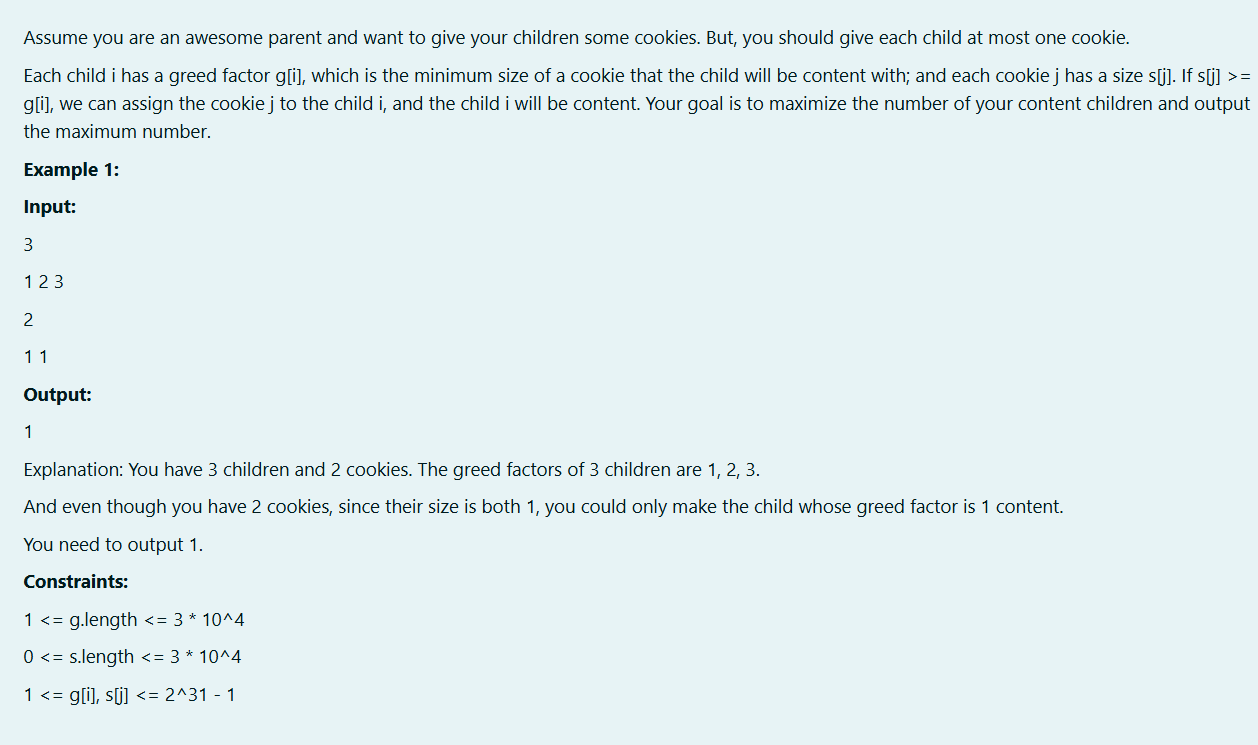
OUTPUT:



RESULT: Thus the program was executed successfully and the output was verified.

EX 3.2

AIM:



SOURCE CODE:

#include<stdio.h>

int main(){

int n,x,m,res=0,y;

scanf("%d",&n);

int children[n];

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

scanf("%d",&x);

children[i] = x;

}

scanf("%d",&m);

int cookies[m];

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

scanf("%d",&x);

cookies[i] = x;

}

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

y = cookies[i];

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

if(children[j] >= y) res++;

children[j] = -1;

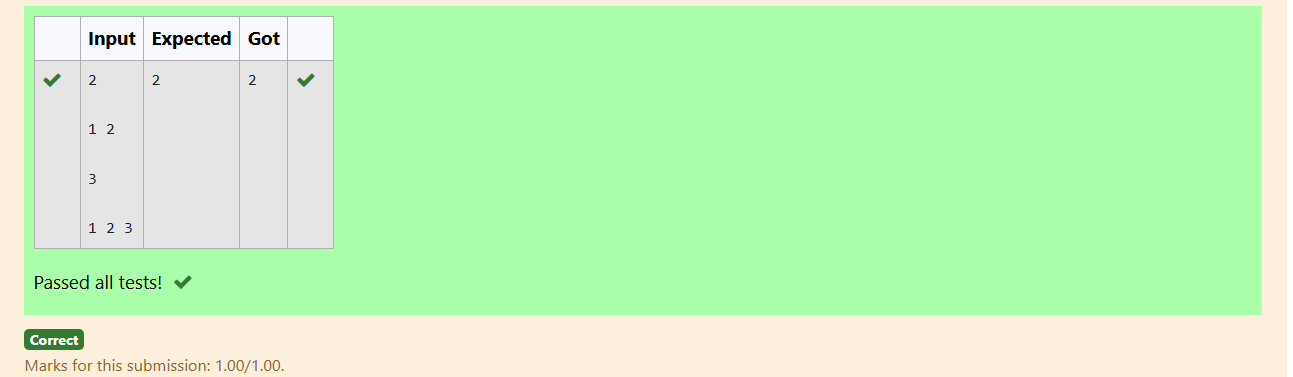
}

}

printf("%d",res);

}

OUTPUT:



RESULT: Thus the program was executed successfully and the output was verified.

EX 4.3

AIM:

#include<stdio.h>

#include<math.h>

int main(){

int n;

scanf("%d",&n);

int a[n],count=0;

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

{

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

}

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

{

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

{

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

int t=a[j];

a[j]=a[j+1];

a[j+1]=t;

}

}

}

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

{

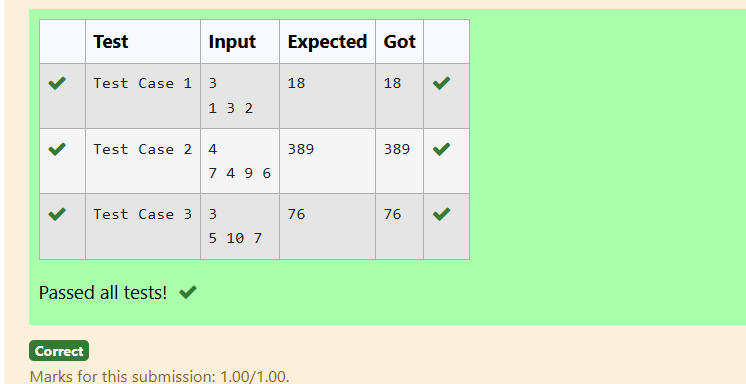
count+=pow(n,i)\*a[i];

}

printf("%d",count) ;

}

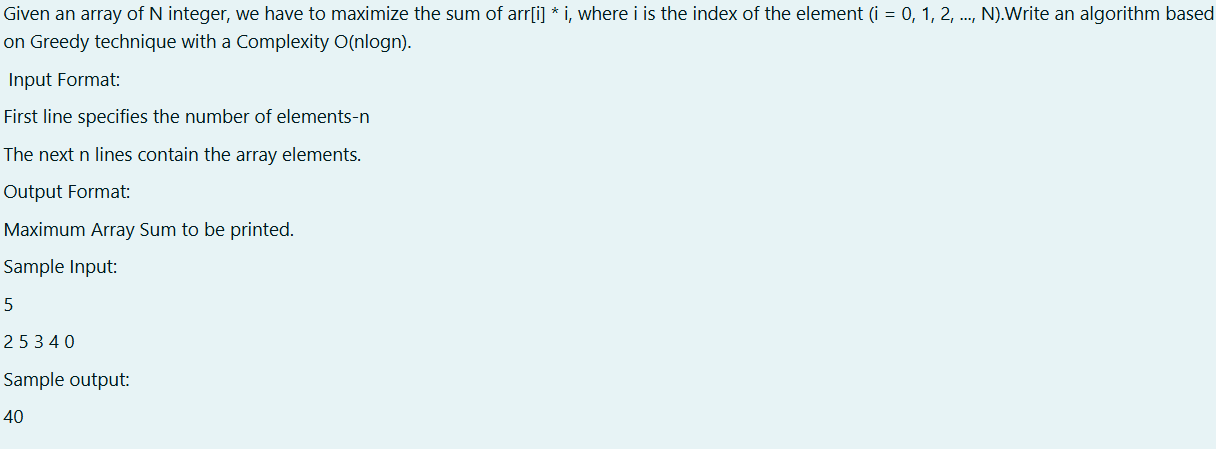
OUTPUT:



RESULT: Thus the program was executed successfully and the output was verified.

EX 4.3

AIM:



SOURCE CODE:

#include<stdio.h>

int main(){

int x,temp,sum=0;

scanf("%d",&x);

int arr[x];

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

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

}

for(int m=0;m<x;m++){

for(int j=0;j<x-1-m;j++){

if(arr[j]>arr[j+1]){

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

for(int l=0;l<x;l++){

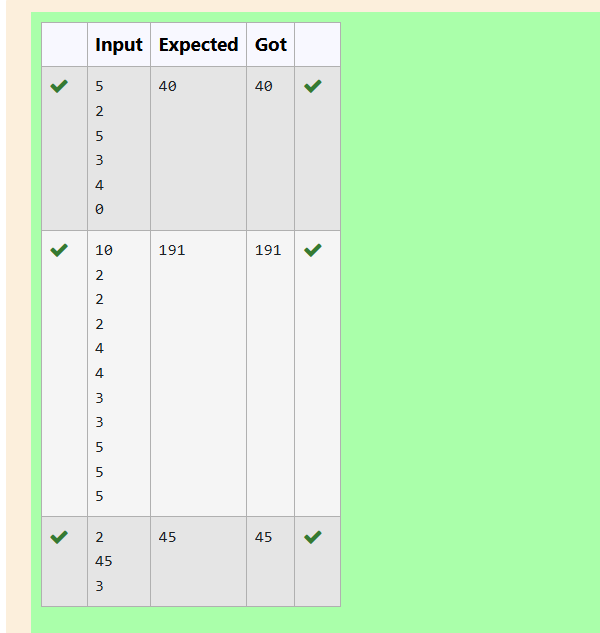
sum=sum+arr[l]\*l;

}

printf("%d",sum);

}

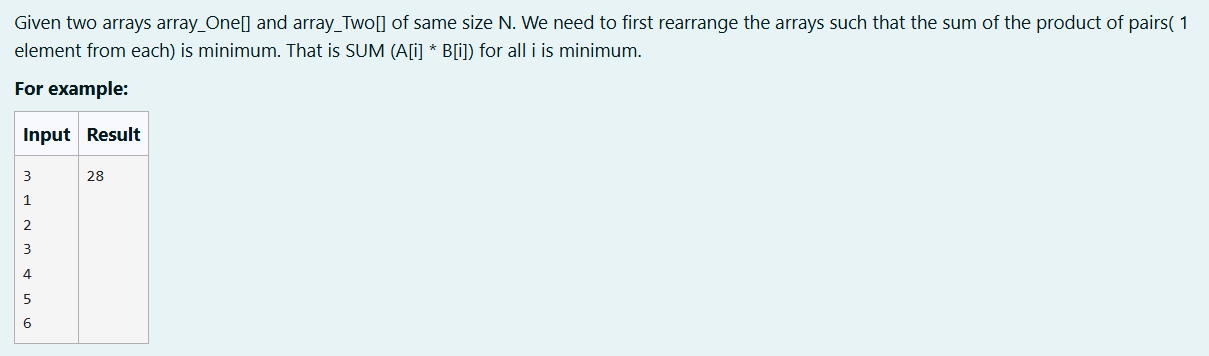
OUTPUT:



RESULT: Thus the program was executed successfully and the output was verified.

EX 3.5

AIM:



SOURCE CODE:

#include<stdio.h>

#include<stdlib.h>

void merge(int arr[], int left, int mid, int right) {

int n1 = mid - left + 1, n2 = right - mid;

int leftArr[n1], rightArr[n2], i, j, k;

for (i = 0; i < n1; i++) leftArr[i] = arr[left + i];

for (j = 0; j < n2; j++) rightArr[j] = arr[mid + 1 + j];

i = 0, j = 0, k = left;

while (i < n1 && j < n2) {

arr[k++] = (leftArr[i] <= rightArr[j]) ? leftArr[i++] : rightArr[j++];

}

while (i < n1) arr[k++] = leftArr[i++];

while (j < n2) arr[k++] = rightArr[j++];

}

void mergeSort(int arr[], int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

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]);

mergeSort(A, 0, n - 1);

mergeSort(B, 0, n - 1);

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

sum += A[i] \* B[j];

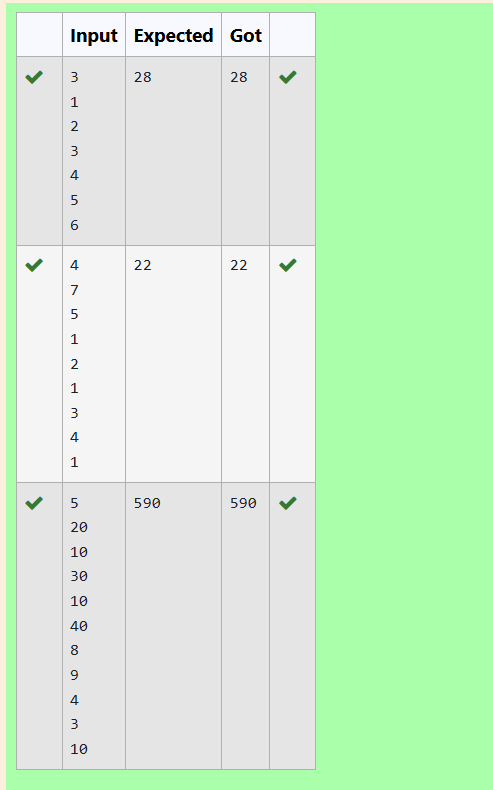
}

printf("%d", sum);

return 0;

}

OUTPUT:



RESULT: Thus the program was executed successfully and the output was verified.