**Week 2**

**GREEDY ALGORITHMS**

**Program 1:**

**Aim:**

To implement a program to determine the minimum number of coins and/or notes required to make change for a given value V using an infinite supply of given denominations.

**Input:**

An integer V, representing the amount to make change for.

**Code:**

#include<stdio.h>

int main(){

int a,total=0;

scanf("%d",&a);

int denom[9]={1000,500,100,50,20,10,5,2,1};

//denom[9]={1,2,5,10,20,50,100,500,1000};

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

if(a>denom[i]){

total+=a/denom[i];

a=a - a/denom[i]\*denom[i];

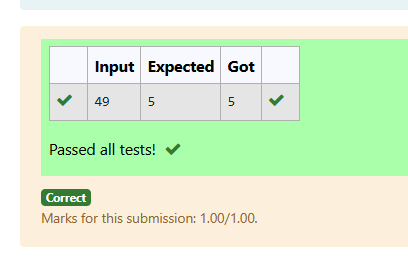
}

}

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

}

**Output:**

****

**Program – 2:**

### ****Aim****:

To implement a program to maximize the number of content children by distributing cookies such that each child gets at most one cookie that satisfies their greed factor.

### ****Input****:

* An integer representing the number of children, n.
* An array of greed factors g (size n).
* An integer representing the number of cookies, m.
* An array of cookie sizes s (size m).

### ****Code:****

#include <stdio.h>

int main(){

int child,cookie,c=0;

scanf("%d",&child);

int children[child];

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

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

}

scanf("%d",&cookie);

int cookies[cookie];

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

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

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

if(children[j]==cookies[i]){

c++;

}

}

}

printf("%d",c);

}

### ****Output:****

### 

**Program – 3:**

**Aim:**

To implement a program that calculates the minimum distance a person needs to run to burn calories after eating burgers in an optimal order using a greedy approach.

**Input:**

* An integer n, representing the number of burgers.
* An array of integers, representing the calorie count of each burger.

**Code:**

#include<stdio.h>

#include<math.h>

int main(){

int a,sum=0,temp=0;

scanf("%d",&a);

int b[a];

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

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

}

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

for(int j=i+1;j<a;j++){

if(b[i]<b[j]){

temp=b[i];

b[i]=b[j];

b[j]=temp;

}

}

}

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

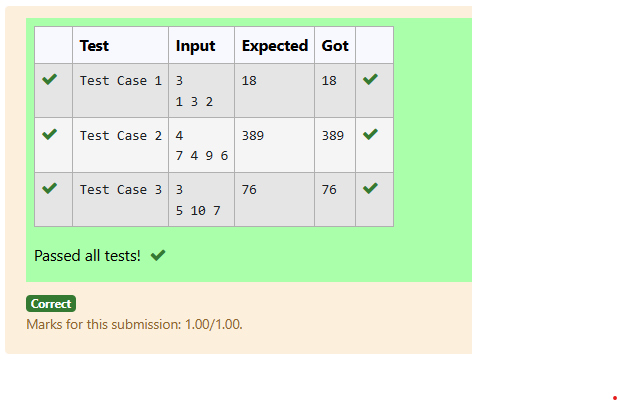
sum+=(int)pow(a,i)\*b[i];

}

printf("%d",sum);

**}**

**Output:**

****

**Program – 4**

**Aim:**

To implement a program to maximize the sum of arr[i]×i\text{arr}[i] \times iarr[i]×i for a given array using a greedy approach with O(nlog⁡n)O(n \log n)O(nlogn) complexity.

**Input:**

* An integer nnn, the number of elements in the array.
* An array of nnn integers.

**Coding:**

#include<stdio.h>

#include<math.h>

int main(){

int a,sum=0,temp=0;

scanf("%d",&a);

int b[a];

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

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

}

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

for(int j=i+1;j<a;j++){

if(b[i]>b[j]){

temp=b[i];

b[i]=b[j];

b[j]=temp;

}

}

}

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

sum+=i\*b[i];

}

printf("%d",sum);

}

**Output:**

****

**Program 5:**

**Aim:**

To implement a program to rearrange two arrays array\_One[] and array\_Two[] of the same size N such that the sum of the product of pairs (i.e., SUM(A[i]×B[i])\text{SUM}(A[i] \times B[i])SUM(A[i]×B[i]) for all iii) is minimized.

**Input:**

* An integer N, the size of both arrays.
* An array array\_One[N] of integers.
* An array array\_Two[N] of integers.

**Code:**

#include<stdio.h>

#include<math.h>

int main(){

int a,sum=0,temp=0;

scanf("%d",&a);

int b[a],c[a];

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

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

}

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

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

}

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

for(int j=i+1;j<a;j++){

if(b[i]<b[j]){

temp=b[i];

b[i]=b[j];

b[j]=temp;

}

}

}

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

for(int j=i+1;j<a;j++){

if(c[i]>c[j]){

temp=c[i];

c[i]=c[j];

c[j]=temp;

}

}

}

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

sum+=c[i]\*b[i];

}

printf("%d",sum);

}

**Output:**

