DAA MOODLE PROGRAMS

GREEDY ALGORITHM PROGRAMS

230701037

Arun Prakash M

CSE-A

1.

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.

CODE-

```c
#include<stdio.h>
int main(){
    int v;
    scanf("%d",&v);
    int d[]={1000,500,100,50,20,10,5,2,1};
    int n=sizeof(d)/sizeof(d[0]);
    int c=0;
    for(int i=0;i<n;i++){
        while(v>=d[i])
        {
        v-=d[i];
        c++;

        }
    }
    printf("%d",c);


}
```

INPUT-

Take an integer from stdin.

OUTPUT-

Print the integer which is change of the number.

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2.

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.

CODE-

```c
#include<stdio.h>
int main()
{
    int x,y,count=0;
    scanf("%d",&x);
    int a[x];
    for(int i=0;i<x;i++)
    {
        scanf("%d",&a[i]);
    }
    scanf("%d",&y);
    int b[y];
    for(int i=0;i<y;i++) {
        scanf("%d",&b[i]);
    }
    for(int i=0;i<y;i++)
    {
        if(a[i]==b[i])
        {
            count++;
        }
    }
    printf("%d",count);
}
```

INPUT-

3

1 2 3

2

1 1

OUTPUT-

1

	Input	Expected	Got	
✔	2 1 2 3 1 2 3	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

3.

AIM-

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.

CODE-

```c
#include<stdio.h>
#include<math.h>
void selectionSort(int arr[], int n)
{
for (int i = 0; i < n - 1; i++)
{
int min_idx = i;

for (int j = i + 1; j < n; j++)
{
if (arr[j] > arr[min_idx])
{
    min_idx = j;
}
}
int temp = arr[min_idx];
arr[min_idx] = arr[i];
arr[i] = temp;
}
}
int main()
{
int n;
scanf("%d",&n);
int arr[n];
for(int i = 0; i< n; i++){
    scanf("%d",&arr[i]);
}
selectionSort(arr,n);
int s = 0;
for(int i = 0; i< n; i++){
    s+= pow(n,i) * arr[i];
}
printf("%d",s);
}
```

INPUT-

First Line contains the number of burgers Second line contains calories of each burger which is n space-separate integers

OUTPUT-

Print: Minimum number of kilometers needed to run to burn out the calories

	Test	Input	Expected	Got	
✔	Test Case 1	3 1 3 2	18	18	✔
✔	Test Case 2	4 7 4 9 6	389	389	✔
✔	Test Case 3	3 5 10 7	76	76	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

4.

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).

CODE-

```c
#include <stdio.h>
#include <stdlib.h>
int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    qsort(arr, n, sizeof(int), compare);
    int maxSum = 0;
    for (int i = 0; i < n; i++) {
        maxSum += arr[i] * i;
    }
    printf("%d\n", maxSum);
}

```

INPUT-

First line specifies the number of elements-n

The next n lines contain the array elements.

OUTPUT-

Maximum Array Sum to be printed.

	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! ✔

5.

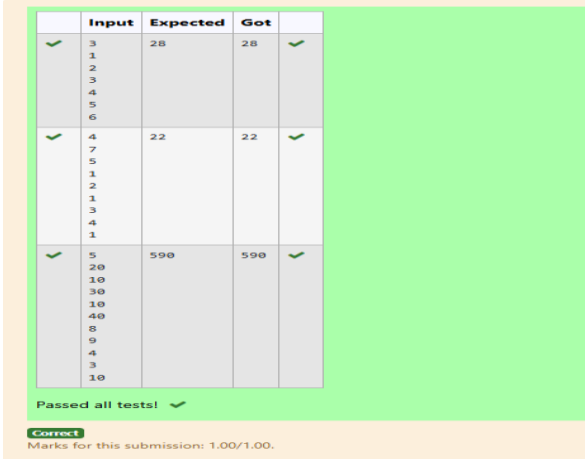
AIM-

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.

CODE-

```c
#include <stdio.h>
void sortArray(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
int main() {
    int n;
    scanf("%d", &n);
    int array_One[n], array_Two[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_One[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_Two[i]);
    }
    sortArray(array_One, n);
    sortArray(array_Two, n);
    int start = 0;
    int end = n - 1;
    while (start < end) {
        int temp = array_Two[start];
        array_Two[start] = array_Two[end];
        array_Two[end] = temp;
        start++;
        end--;
    }
    int minSum = 0;
    for (int i = 0; i < n; i++) {
        minSum += array_One[i] * array_Two[i];
    }
    printf("%d\n", minSum);
}
```

OUTPUT-



	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.