DAA MOODLE PROGRAMS GREEDY ALGORITHM PROGRAMS

230701014

Adithyaa Suresh

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-

```
#include<stdio.h>
 2 v int main(){
        int v;
scanf("%d",&v);
 3
 4
         int d[]=\{1000,500,100,50,20,10,5,2,1\};
 5
 6
         int n=sizeof(d)/sizeof(d[0]);
         int c=0;
 7
 8
         for(int i=0;i<n;i++){</pre>
 9
             while(v>=d[i])
10
11
             v-=d[i];
12
             C++;
13
14
             }
15
         printf("%d",c);
16
17
18
19
    }
20
```

INPUT-

Take an integer from stdin.

OUTPUT-

Print the integer which is change of the number.



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-

```
#include<stdio.h>
     int main()
 2
 3
           int x,y,count=0;
scanf("%d",&x);
int a[x];
 4
 5
 6
           for(int i=0;i<x;i++)</pre>
 8
                scanf("%d",&a[i]);
 9
10
11
           scanf("%d",&y);
           int b[y];
for(int i=0;i<y;i++) {
    scanf("%d",&b[i]);</pre>
12
13
14
15
16
           for(int i=0;i<y;i++)
17
18
                if(a[i]==b[i])
19
                {
20
                      count++:
21
                }
22
           printf("%d",count);
23
    13
24
```

INPUT-

3

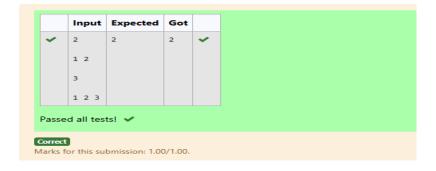
123

2

11

OUTPUT-

1



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-

```
#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])
10 v
11
12 v
13
14
15
16
17
18
19
20
21
22 v
             min_idx = j;
      int temp = arr[min_idx];
arr[min_idx] = arr[i];
arr[i] = temp;
        int main()
      24
25
26 v
27
28
29
30
31 v
      stan.;
}
selectionSort(arr,n);
int s = 0;
for(int i = 0; i < n; i++){
    s+= pow(n,i) * arr[i];</pre>
33
34
35
36
37
      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



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-

```
#include <stdio.h>
     #include <stdlib.h>
     int compare(const void *a, const void *b) {
   return (*(int*)a - *(int*)b);
 3 ,
 4
 5
 6 ,
     int main() {
          int n;
 7
           scanf("%d", &n);
 8
           int arr[n];
 9
          for (int i = 0; i < n; i++) {
10 •
               scanf("%d", &arr[i]);
11
12
          gsort(arr, n, sizeof(int), compare);
int maxSum = 0;
for (int i = 0; i < n; i++) {</pre>
13
14
15 ,
16
               maxSum += arr[i] * i;
17
          printf("%d\n", maxSum);
18
19
20
```

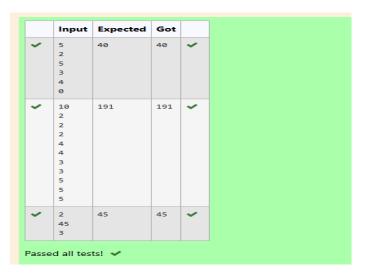
INPUT-

First line specifies the number of elements-n

The next n lines contain the array elements.

OUTPUT-

Maximum Array Sum to be printed.



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-

```
#include <stdio.h>
         pinclude <std10.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;
        }
}
   6
   8
   9
10
                              }
11
12
          int main() {
13
                   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]);
}</pre>
14
15
16
17
18
19
                    for (int i = 0; i < n; i++) {
    scanf("%d", &array_Two[i]);</pre>
20
21
22
23
                    sortArray(array_One, n);
sortArray(array_Two, n);
                    int start = 0;
int end = n - 1;
while (start < end) {</pre>
25
26
27
                            int temp = array_Two[start];
array_Two[start] = array_Two[end];
array_Two[end] = temp;
start++;
28
29
30
31
32
                              end--;
33
                    int minSum = 0;
for (int i = 0; i < n; i++) {
    minSum += array_One[i] * array_Two[i];</pre>
34
35
36
37
                    printf("%d\n", minSum);
39
```

OUTPUT-

