GREEDY ALGORITHMS

1)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.

Answer:

```
#include <stdio.h>

int minCoins(int V) {

    int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int n = sizeof(denominations) / sizeof(denominations[0]);
    int count = 0;


    for (int i = 0; i < n; i++) {
        if (V == 0) {
            break;
        }

        count += V / denominations[i];

        V = V % denominations[i];
    }

    return count;
}
```

GREEDY ALGORITHMS

```
int main() {
    int V;



    scanf("%d", &V);

    int result = minCoins(V);
    printf("%d\n", result);

    return 0;
}
```

	Input	Expected	Got	
	49	5	5	

Passed all tests!

2)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.

You need to output 1.

Constraints:

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1

Answer

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int num_children, num_cookies;

    scanf("%d", &num_children);
    int greed_factors[num_children];

    for (int i = 0; i < num_children; i++) {
        scanf("%d", &greed_factors[i]);
    }

    scanf("%d", &num_cookies);
    int cookie_sizes[num_cookies];

    for (int i = 0; i < num_cookies; i++) {
        scanf("%d", &cookie_sizes[i]);
    }

    for (int i = 0; i < num_children - 1; i++) {
        for (int j = 0; j < num_children - i - 1; j++) {
            if (greed_factors[j] > greed_factors[j + 1]) {
                int temp = greed_factors[j];
                greed_factors[j] = greed_factors[j + 1];
                greed_factors[j + 1] = temp;
            }
        }
    }
for (int i = 0; i < num_cookies - 1; i++) {
        for (int j = 0; j < num_cookies - i - 1; j++) {
            if (cookie_sizes[j] > cookie_sizes[j + 1]) {
                int temp = cookie_sizes[j];
                cookie_sizes[j] = cookie_sizes[j + 1];
                cookie_sizes[j + 1] = temp;
            }
        }
    }

    int child_index = 0;
    int cookie_index = 0;
    int content_children = 0;

    while (child_index < num_children && cookie_index < num_cookies) {
        if (cookie_sizes[cookie_index] >= greed_factors[child_index]) {
            // Cookie can satisfy the child
            content_children++;
            child_index++;
        }
        cookie_index++;
    }
    printf("%d\n", content_children);

    return 0;
}
```

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

Passed all tests!

3)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 $3_i * c$ 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 (3_0 * 1) + (3_1 * 3) + (3_2 * 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.

Input Format

First Line contains the number of burgers

Second line contains calories of each burger which is n space-separate 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
```

For example:

Test	Input	Result
Test Case 1	3 1 3 2	18

Answer:

```
#include<stdio.h>
#include<math.h>
int main(){
   int n,a[50];
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d",&a[i]);
   }
   int temp;
   for(int i=0;i<n;i++){
      for(int j=i+1;j<n;j++){
       if(a[j]>a[i]){
          temp=a[i];
          a[i]=a[j];
```

```
            a[j]=temp;
        }
       }
   }
   int sum=0;
   for(int i=0;i<n;i++){
       sum+=pow(n,i)*a[i];
   }
   printf("%d",sum);
}
```

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

4)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.

GREEDY ALGORITHMS

Sample Input:

5

2 5 3 4 0

Sample output:

40

Answer

```
#include<stdio.h>
int main(){
  int sum=0;
  int a[50],n;
  scanf("%d",&n);
  for(int i=0;i<n;i++){
     scanf("%d",&a[i]);
  }
  int temp;
 for(int i=0;i<n;i++){
    for(int j=i+1;j<n;j++){
      if(a[j]<a[i]){
        temp=a[j];
        a[j]=a[i];
        a[i]=temp;
      }
    }
  }

  for(int i=0;i<n;i++){
    sum+=a[i]*i;
  }
  printf("%d",sum);
}
```

Input	Expected	Got	

GREEDY ALGORITHMS

	5 2 5 3 4 0	40	40	
	10 2 2 2 4 4 3 3 5 5 5	191	19 1	
	2 45 3	45	45	

Passed all tests!

5)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

Answer:

```
#include <stdio.h>
#include <stdlib.h>
```

GREEDY ALGORITHMS

```
int compareAsc(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

int compareDesc(const void *a, const void *b) {
    return (*(int *)b - *(int *)a);
}

int minimumProductSum(int *array_One, int *array_Two, int size) {
    qsort(array_One, size, sizeof(int), compareAsc);
    qsort(array_Two, size, sizeof(int), compareDesc);

    int min_sum = 0;
    for (int i = 0; i < size; i++) {
        min_sum += array_One[i] * array_Two[i];
    }

    return min_sum;
}

int main() {
    int N;


    scanf("%d", &N);

    int *array_One = (int *)malloc(N * sizeof(int));
    int *array_Two = (int *)malloc(N * sizeof(int));


    for (int i = 0; i < N; i++) {
        scanf("%d", &array_One[i]);
    }


    for (int i = 0; i < N; i++) {
        scanf("%d", &array_Two[i]);
    }
```

GREEDY ALGORITHMS

```
    int result = minimumProductSum(array_One, array_Two, N);
    printf( "%d\n", result);

    free(array_One);
    free(array_Two);

    return 0;
}
```

	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	