Design and Analysis of Algorithms

CS23331

Sujit P
240701544

EXPERIMENT 1

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void function (int n)
{
    int i= 1;

    int s =1;

    while(s <= n)
    {
        i++;
        s += i;
     }
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:

Print the value of the counter variable

For example:

Input	Result
9	12

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
int c = 0,n;
int function (int n)
{

    int i= 1;
    int s =1;
    c+=2;
    while(s <= n)
    {
        c++;
        i++;
        c++;
        s += i;
        c++;
    }
    c++;

    return c;
}

int main(){

    int n;
    scanf("%d",&n);
    function(n);


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

	Input	Expected	Got	
✔	9	12	12	✔
✔	4	9	9	✔

EXPERIMENT 2

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void func(int n)
{
    if(n==1)
    {
      printf("*");
    }
    else
    {
     for(int i=1; i<=n; i++)
     {
       for(int j=1; j<=n; j++)
       {
          printf("*");
          printf("*");
          break;
       }
     }
   }
 }
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer: (penalty regime: 0 %)

```
#include <stdio.h>

int c;

int func(int n)
{

    if(n==1)
    {
        c++;
        printf("*");
        c++;
    }


    else
    { c++;

        for(int i=1; i<=n; i++)
        {
            c++;

            for(int j=1; j<=n; j++)
            {

                c++;
                //printf("*");
                c++;
                //printf("*");
                c++;
                break;
            }
            c++;
        }
        c++;
    }
    return c;
}


int main(){
    int n,d;
    scanf("%d",&n);

    d = func(n);
    printf("%d",d);

}
```

	Input	Expected	Got	
✔	2	12	12	✔
✔	1000	5002	5002	✔
✔	143	717	717	✔

Passed all tests! ✔

Correct

EXPERIMENT 3

Convert the following algorithm into a program and find its time complexity using counter method.

```
Factor(num) {
{
   for (i = 1; i <= num;++i)
   {
    if (num % i== 0)
       {
         printf("%d ", i);
       }
    }
 }
```

Note: No need of counter increment for declarations and scanf() and counter variable printf() statement.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

```c
#include <stdio.h>
int c = 0 ;
int Factor(int num)
{
    for (int i = 1; i <= num;++i)
    {
        c++;
        if (num % i== 0)
        {
            c++;
            //printf("%d ", i);

        }
        c++;

    }
    c++;
    return c;
}

int main(){
    int n;
    scanf("%d",&n);
    Factor(n);
    printf("%d",c);
}
```

	Input	Expected	Got	
✔	12	31	31	✔
✔	25	54	54	✔
✔	4	12	12	✔

Passed all tests! ✔

Correct

EXPERIMENT 4

Convert the following algorithm into a program and find its time

complexity using counter method.

```
void function(int n)
{
    int c= 0;
    for(int i=n/2; i<n; i++)
        for(int j=1; j<n; j = 2 * j)
            for(int k=1; k<n; k = k * 2)
                c++;
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

```c
#include <stdio.h>

int function(int n)
{
    int c= 0;
    c++;
    for(int i=n/2; i<n; i++){
        c+=2;
        for(int j=1; j<n; j = 2 * j){
            c+=2;
            for(int k=1; k<n; k = k * 2){
                c+=2;

            }


        }

    }
    c++;
    return c;
}

int main(){
    int n;
    scanf("%d",&n );
    printf("%d",function(n));
}
```

	Input	Expected	Got	
✔	4	38	38	✔
✔	10	212	212	✔

Passed all tests! ✔

Correct

EXPERIMENT 5

Convert the following algorithm into a program and find its time complexity using counter method.

```
void reverse(int n)
{
   int rev = 0, remainder;
   while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n/= 10;

    }
print(rev);
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

```c
#include <stdio.h>

int reverse(int n)
{
    int rev = 0,c = 0, rem;
    c++;
    while (n != 0)
    {
        c++;
        rem = n % 10;
        c++;
        rev = rev * 10 + rem;
        c++;
        n/= 10;
        c++;

    }
    c++;
    c++;
    return c;

    //printf(rev);

}

int main(){
    int n;
    scanf("%d",&n);
    printf("%d",reverse(n));
}
```

	Input	Expected	Got	
✔	12	11	11	✔
✔	1234	19	19	✔

Passed all tests! ✔

EXPERIMENT 6

Problem Statement

Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using Divide and Conquer to Count the number of zeroes in the given array.

Input Format

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

Output Format

First Line Contains Integer – Number of zeroes present in the given array.

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main(){
    int m,a[100];
    scanf("%d",&m);
    for(int i=0;i<m;i++){
        scanf("%d",&a[i]);
    }
    int c=0;
    for(int i=0;i<m;i++){
        if(a[i]==1){
            c++;
        }
    }
    printf("%d",m-c);
}
```

	Input	Expected	Got	
✔	5 1 1 1 0 0	2	2	✔
✔	10 1 1 1 1 1 1 1 1 1 1	0	0	✔

EXPERIMENT 7

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than `⌊n / 2⌋` times. You may assume that the majority element always exists in the array.

Example 1:

```
Input: nums = [3,2,3]
Output: 3
```

Example 2:

```
Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

Constraints:

- `n == nums.length`
- `1 <= n <= 5 * 10^4`
- `-2^31 <= nums[i] <= 2^31 - 1`

For example:

Input	Result
3 3 2 3	3
7 2 2 1 1 1 2 2	2

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int countel(int x , int a[100],int m){
    int c = 0;
    for(int i=0;i<m;i++){
        if(a[i]==x){
            c++;
        }
    }
    return c;
}
int main(){
    int m,a[100];
    scanf("%d",&m);
    for(int i=0;i<m;i++){
        scanf("%d",&a[i]);
    }
    for(int i=0;i<m;i++){
        if(countel(a[i],a,m) > m/2){
            printf("%d",a[i]);
            break;
        }
    }

}
```

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Correct

EXPERIMENT 8

Problem Statement:

Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Value for x

Output Format

First Line Contains Integer – Floor value for x

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findFloor(int arr[], int n, int x) {
    int left = 0, right = n - 1, res = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            return arr[mid];
        } else if (arr[mid] < x) {
            res = arr[mid];
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return res;
}

int main() {
    int n, x;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    scanf("%d", &x);
    int floor = findFloor(arr, n, x);
    printf("%d\n", floor);
    return 0;
}
```

	Input	Expected	Got	
✔	6 1 2 8 10 12 19 5	2	2	✔
✔	5 10 22 85 108 129 100	85	85	✔

EXPERIMENT 9

Problem Statement:

Given a sorted array of integers say arr[] and a number x. Write a recursive program using divide and conquer strategy to check if there exist two elements in the array whose sum = x. If there exist such two elements then return the numbers, otherwise print as "No".

Note: Write a Divide and Conquer Solution

Input Format

First Line Contains Integer n – Size of array

Next n lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Sum Value

Output Format

First Line Contains Integer – Element1

Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findPair(int arr[] , int left , int right , int x, int *a , int *b){
    if(left >= right){
        return 0;
    }

    int sum = arr[left] + arr[right];

    if(sum == x){
        *a = arr[left];
        *b = arr[right];
        return 1;
    }

    else if(sum < x)
        return findPair(arr,left+1,right,x,a,b);

    else if(sum > x)
        return findPair(arr,left,right-1,x,a,b);


    return 0;
}

int main(){
    int n,x;
    scanf("%d",&n);

    int arr[n];

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

    scanf("%d",&x);

    int a , b;

    if(findPair(arr,0,n-1,x,&a,&b)){
        printf("%d\n%d",a,b);
    }
    else{
        printf("No\n");
    }
}
```

	Input	Expected	Got	
✔	4 2 4 8 10 14	4 10	4 10	✔

EXPERIMENT 10

Write a Program to Implement the Quick Sort Algorithm

Input Format:
The first line contains the no of elements in the list-n
The next n lines contain the elements.

Output:
Sorted list of elements

For example:

Input	Result
5 67 34 12 98 78	12 34 67 78 98

Answer:

```c
#include <stdio.h>

// Function to swap two elements
void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

// Partition function
int partition(int arr[], int low, int high) {
    int pivot = arr[high]; // pivot
    int i = (low - 1); // Index of smaller element

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++; // increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

// Quick Sort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];

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

    quickSort(arr, 0, n - 1);

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

	Input	Expected	Got	
✔	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✔

EXPERIMENT 11

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: (penalty regime: 0 %)

```c
#include <stdio.h>

int main(){
    int v;
    scanf("%d",&v);

    int th = v/1000;
    int fh = (v%1000)/500;
    int hd = ((v%1000)%500)/100;
    int ft = (((v%1000)%500)%100)/50;
    int ty = ((((v%1000)%500)%100)%50)/20;
    int tn = (((((v%1000)%500)%100)%50)%20)/10;
    int fv = ((((((v%1000)%500)%100)%50)%20)%10)/5;
    int tw = (((((((v%1000)%500)%100)%50)%20)%10)%5)/2;
    int on = ((((((((v%1000)%500)%100)%50)%20)%10)%5)%2;

    int sum = th + fh + hd + ft +ty+ tn+ fv + tw+ on;
    printf("%d",sum);
}
```

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

EXPERIMENT 12

Answer: (penalty regime: 0 %)

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

// Comparison function for qsort
int cmp(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}

int main() {
    int n, m;
    scanf("%d", &n);
    int g[n];
    for (int i = 0; i < n; ++i)
        scanf("%d", &g[i]);
    scanf("%d", &m);
    int s[m];
    for (int i = 0; i < m; ++i)
        scanf("%d", &s[i]);

    // Sort greed and size arrays
    qsort(g, n, sizeof(int), cmp);
    qsort(s, m, sizeof(int), cmp);

    int i = 0, j = 0, result = 0;
    while (i < n && j < m) {
        if (s[j] >= g[i]) {
            // Cookie satisfies this child's greed
            result++;
            i++;
            j++;
        } else {
            // Cookie too small, try next
            j++;
        }
    }
    printf("%d\n", result);
    return 0;
}

```

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

Passed all tests! ✔

EXPERIMENT 13

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: (penalty regime: 0 %)

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

// Comparator for sorting in descending order
int cmp(const void *a, const void *b) {
    return (*(int*)b) - (*(int*)a);
}

int main() {
    int n;
    scanf("%d", &n);
    int cals[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &cals[i]);
    }
    // Descending order sort
    qsort(cals, n, sizeof(int), cmp);

    long long total = 0;
    long long multiplier = 1; // 3^0 = 1 for first burger
    for (int i = 0; i < n; i++) {
        total += ((long long)cals[i]) * multiplier;
        multiplier *= 3;
    }
    printf("%lld\n", total);
    return 0;
}
```

EXPERIMENT 14

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.

Sample Input:

5

2 5 3 4 0

Sample output:

40

Answer: (penalty regime: 0 %)

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

int cmp(const void *a, const void *b) {
    return (*(int*)a) - (*(int*)b); // ascending
}

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), cmp);
    long long sum = 0;
    for(int i = 0; i < n; i++)
        sum += (long long)arr[i] * i;
    printf("%lld\n", sum);
    return 0;
}
```

	Input	Expected	Got	
✔	5 2 5 3 4 0	40	40	✔
✔	10 2 2 2 4 4 3 3 5 5 5	191	191	✔

EXPERIMENT 15

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: (penalty regime: 0 %)

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

// Comparison functions for qsort
int asc(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
int desc(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

int main() {
    int n;
    scanf("%d", &n); // Input array size

    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]);

    // Sort first array in ascending
    qsort(array_One, n, sizeof(int), asc);
    // Sort second array in descending
    qsort(array_Two, n, sizeof(int), desc);

    int result = 0;
    for(int i = 0; i < n; i++)
        result += array_One[i] * array_Two[i];

    printf("%d\n", result);
    return 0;
}
```

	Input	Expected	Got	
✔	3 1 2 3 4 5 6	28	28	✔

EXPERIMENT 16

Playing with Numbers:

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3.Write any efficient algorithm to find the possible ways.

Example 1:

Input: *6*
Output:*6*
Explanation: *There are 6 ways to 6 represent number with 1 and 3*

1+1+1+1+1+1
3+3
1+1+1+3
1+1+3+1
1+3+1+1
3+1+1+1

Input Format

First Line contains the number n

Output Format

Print: The number of possible ways 'n' can be represented using 1 and 3

Sample Input

6

Sample Output

6

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    long long dp[n+1]; // Use long long for large cases
    dp[0] = 1;
    for (int i = 1; i <= n; i++) {
        dp[i] = 0;
        dp[i] += dp[i-1];
        if (i >= 3) dp[i] += dp[i-3];
    }
    printf("%lld\n", dp[n]);
    return 0;
}
```

	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	24382819596721629	24382819596721629	✔

Passed all tests! ✔

EXPERIMENT 17

Playing with Chessboard:

Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

Example:

Input

3

1 2 4

2 3 4

8 7 1

Output:

19

Explanation:

Totally there will be 6 paths among that the optimal is

Optimal path value:1+2+8+7+1=19

Input Format

First Line contains the integer n

The next n lines contain the n*n chessboard values

Output Format

Print Maximum monetary value of the path

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

#define MAX_N 100

int main() {
    int n;
    int chessboard[MAX_N][MAX_N];
    int dp[MAX_N][MAX_N];


    scanf("%d", &n);


    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &chessboard[i][j]);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == 0 && j == 0) {
                dp[i][j] = chessboard[i][j];
            } else if (i == 0) {
                dp[i][j] = dp[i][j-1] + chessboard[i][j];
            } else if (j == 0) {
                dp[i][j] = dp[i-1][j] + chessboard[i][j];
            } else {
                if (dp[i-1][j] > dp[i][j-1])
                    dp[i][j] = dp[i-1][j] + chessboard[i][j];
                else
                    dp[i][j] = dp[i][j-1] + chessboard[i][j];
            }
        }
    }


    printf("%d\n", dp[n-1][n-1]);

    return 0;
}
```

EXPERIMENT 18

Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

s1: ggtabe

s2: tgatasb

s1	a	g	g	t	a	b	
s2	g	x	t	x	a	y	b

The length is 4

Solveing it using Dynamic Programming

For example:

Input	Result
aab azb	2

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <string.h>

int max(int a, int b) {
    return (a > b) ? a : b;
}

int lcs(char* X, char* Y, int m, int n) {
    int dp[m + 1][n + 1];
    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (X[i - 1] == Y[j - 1])
                dp[i][j] = 1 + dp[i - 1][j - 1];
            else
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    }
    return dp[m][n];
}

int main() {
    char X[100], Y[100];

    scanf("%s", X);

    scanf("%s", Y);

    int m = strlen(X);
    int n = strlen(Y);

    printf("%d\n", lcs(X, Y, m, n));
    return 0;
}
```

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

EXPERIMENT 19

Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Input:9

Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int longestNonDecreasingSubsequence(int arr[], int n) {
    int dp[n];
    for(int i = 0; i < n; i++)
        dp[i] = 1;

    for(int i = 1; i < n; i++) {
        for(int j = 0; j < i; j++) {
            if(arr[i] >= arr[j] && dp[i] < dp[j]+1)
                dp[i] = dp[j]+1;
        }
    }

    // Find the maximum in dp[]
    int max = dp[0];
    for(int i = 1; i < n; i++) {
        if(dp[i] > max)
            max = dp[i];
    }
    return max;
}

int main() {
    int arr[] = {-1,3,4,5,2,2,2,2,3};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("%d\n", longestNonDecreasingSubsequence(arr, n));
    return 0;
}
```

	Input	Expected	Got	
✔	9 -1 3 4 5 2 2 2 2 3	6	6	✔
✔	7 1 2 2 4 5 7 6	6	6	✔

Passed all tests! ✔

EXPERIMENT 20

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int count(int arr[] , int n , int x){
    int c = 0;
    for(int i = 0 ; i<n ; i++){
        if(arr[i] == x){
            c++;
        }
    }
    return c;
}

int main(){
    int n ;
    scanf("%d",&n);
    int arr[n];
    for(int i = 0 ; i<n ; i++){
        scanf("%d",&arr[i]);
    }
    for(int i = 0 ; i<n ; i++){
        if(count(arr,n,arr[i]) > 1){
            printf("%d",arr[i]);
            break;
        }

    }

}
```

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

EXPERIMENT 21

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int count(int arr[] , int n , int x){
    int c = 0;
    for(int i = 0 ; i<n ; i++){
        if(arr[i] == x){
            c++;
        }
    }
    return c;
}

int main(){
    int n ;
    scanf("%d",&n);
    int arr[n];
    for(int i = 0 ; i<n ; i++){
        scanf("%d",&arr[i]);
    }
    for(int i = 0 ; i<n ; i++){
        if(count(arr,n,arr[i]) > 1){
            printf("%d",arr[i]);
            break;
        }

    }

}
```

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

EXPERIMENT 22

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

For example:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main(){
    int t;
    scanf("%d",&t);
    for(int k = 0 ; k<1 ; k++){
        int m;
        scanf("%d" , &m);
        int a1[m];
        for(int i = 0 ; i<m ; i++){
            scanf("%d",&a1[i]);
        }
        int n;
        scanf("%d", &n);
        int a2[n];
        for(int i = 0 ; i<n ; i++){
            scanf("%d",&a2[i]);
        }

        for(int i = 0;i<m;i++){
            for(int j = 0;j<n;j++){
                if(a1[i] == a2[j]){
                    printf("%d " , a1[i]);
                }
            }
        }

    }
}
```

EXPERIMENT 23

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

For example:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main(){
    int t;
    scanf("%d",&t);
    for(int k = 0 ; k<1 ; k++){
        int m;
        scanf("%d" , &m);
        int a1[m];
        for(int i = 0 ; i<m ; i++){
            scanf("%d",&a1[i]);
        }
        int n;
        scanf("%d", &n);
        int a2[n];
        for(int i = 0 ; i<n ; i++){
            scanf("%d",&a2[i]);
        }

        for(int i = 0;i<m;i++){
            for(int j = 0;j<n;j++){
                if(a1[i] == a2[j]){
                    printf("%d " , a1[i]);
                }
            }
        }

    }
}
```

EXPERIMENT 24

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

Input	Result
3 1 3 5 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main(){
    int m;
    scanf("%d",&m);
    int a[m];

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

    for(int i = 0 ; i<m ; i++){
        for(int j = 0; j<m ; j++){
            if(i!=j && (a[i] - a[j] == n)){
                printf("%d",1);
                return 0;
            }


        }
    }
    printf("%d",0);
    return 0;

}
```

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔

EXPERIMENT 25

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

Input	Result
3 1 3 5 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main(){
    int m;
    scanf("%d",&m);
    int a[m];

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

    for(int i = 0 ; i<m ; i++){
        for(int j = 0; j<m ; j++){
            if(i!=j && (a[i] - a[j] == n)){
                printf("%d",1);
                return 0;
            }


        }
    }
    printf("%d",0);
    return 0;

}
```

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔