RAJALAKSHMI ENGINEERING COLLEGE
RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23331
DESIGN AND ANALYSIS OF ALGORITHMSLAB

Laboratory Observation Note Book

Name : DONEESWARAN J

Year / Branch / Section : 2nd Year/ AIML / A

Register No. 231501502

Semester : 2nd Semester

Academic Year : 2024-2025

TABLE OF INDEX

BASIC C PROGRAMMING PRACTICE

SWAP NUMBERS

Given two numbers, write a C program to swap the given numbers.

For example:

Input	Result
10 20	20 10

PROGRAM

```
#include<stdio.h>
int main()
{
int a=0,b,c;
scanf("%d %d",&b,&c);
a=b;
b=c;
c=a;
printf("%d %d",b,c);
}
```

OUTPUT

	Input	Expected	Got	
	10 20	20 10	20 10	

Passed all tests!

ELIGIBILITY OF ADMISSION

Write a C program to find the eligibility of admission for a professional course based on the following criteria:

Marks in Maths >= 65
Marks in Physics >= 55
Marks in Chemistry >= 50
Or
Total in all three subjects >= 180

Sample Test Cases

Test Case 1

Input
70 60 80
Output
The candidate is eligible

Test Case 2

Input
50 80 80
Output
The candidate is eligible

Test Case 3

Input
50 60 40
Output
The candidate is not eligible

PROGRAM

```
#include<stdio.h>
int main()
{
   int maths,physics,chemistry,subject;
   scanf("%d %d %d",&maths,&physics,&chemistry);
   subject=maths+physics+chemistry;
   if(maths>=65&&physics>=55&&chemistry>=50)
```

```
        {
          printf("The candidate is eligible");
        }
        else if (subject>=180)
        {
          printf("The candidate is eligible");
        }
        else
        {
           printf("The candidate is not eligible");
        }
    }
```

OUTPUT

	Input	Expected	Got	
	70 60 80	The candidate is eligible	The candidate is eligible	
	50 80 80	The candidate is eligible	The candidate is eligible	

Passed all tests!

SUPER MARKET DISCOUNT

Malini goes to BestSave hyper market to buy grocery items. BestSave hyper market provides 10% discount on the bill amount B when ever the bill amount B is more than Rs.2000.
The bill amount B is passed as the input to the program. The program must print the final amount A payable by Malini.

Input Format:
The first line denotes the value of B.

Output Format:
The first line contains the value of the final payable amount A.

Example Input/Output 1:
Input:
1900
Output:
1900

Example Input/Output 2:
Input:
3000
Output:
2700

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b;
   scanf("%d",&a);
   if(a>2000)
   {
      b=a-(0.10*a);
      printf("%d",b);
   }
   else
   {
      printf("%d",a);
   }
}
```

PROGRAM

	Input	Expected	Got	
	1900	1900	1900	
	3000	2700	2700	

Passed all tests!

DONATION

Baba is very kind to beggars and every day Baba donates half of the amount he has when ever a beggar requests him. The money M left in Baba's hand is passed as the input and the number of beggars B who received the alms are passed as the input. The program must print the money Baba had in the beginning of the day.

Input Format:
The first line denotes the value of M.
The second line denotes the value of B.
Output Format:
The first line denotes the value of money with Baba in the beginning of the day.

Example Input/Output:
Input:
100
2
Output:
400
Explanation:
Baba donated to two beggars. So when he encountered second beggar he had 100*2 = Rs.200 and when he encountered 1st he had 200*2 = Rs.400.

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b;
   scanf("%d",&a);
   scanf("%d",&b);
   printf("%d",a*b*2);
}
```

PROGRAM

	Input	Expected	Got	
	100 2	400	400	

Passed all tests!

PUNCTUALITY INCENTIVE

The CEO of company ABC Inc wanted to encourage the employees coming on time to the office. So he announced that for every consecutive day an employee comes on time in a week (starting from Monday to Saturday), he will be awarded Rs.200 more than the previous day as "Punctuality Incentive". The incentive I for the starting day (ie on Monday) is passed as the input to the program. The number of days N an employee came on time consecutively starting from Monday is also passed as the input. The program must calculate and print the "Punctuality Incentive" P of the employee.

Input Format:
The first line denotes the value of I.
The second line denotes the value of N.
Output Format:
The first line denotes the value of P.

Example Input/Output:
Input:
500
3
Output:
2100
Explanation:
On Monday the employee receives Rs.500, on Tuesday Rs.700, on Wednesday Rs.900
So total = Rs.2100

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b,c=0,i;
   scanf("%d",&a);
   scanf("%d",&b);
   for(i=0;i<b;i++)
   {
      c+=a;
      a=a+200;
   }
   printf("%d",c);
}
```

PROGRAM

	Input	Expected	Got	
	5003	2100	2100	
	1003	900	900	

Passed all tests!

DIVISIBLE NUMBER

Two numbers M and N are passed as the input. A number X is also passed as the input. The program must print the numbers divisible by X from N to M (inclusive of M and N).

Input Format:
The first line denotes the value of M
The second line denotes the value of N
The third line denotes the value of X
Output Format:
Numbers divisible by X from N to M, with each number separated by a space.

Boundary Conditions:
1 <= M <= 9999999
M < N <= 9999999
1 <= X <= 9999
Example Input/Output 1:
Input:
2
40
7
Output:
35 28 21 14 7

Example Input/Output 2:
Input:
66
121
11
Output:
121 110 99 88 77 66

PROGRAM

```
#include<stdio.h>
int main()
{
   int m,n,x,i;
   scanf("%d",&m);
   scanf("%d",&n);
   scanf("%d",&x);
   for(i=n;i>m-1;i--)
   {
```

```
            if(i%x==0)
            {
             printf("%d ",i);
            }
        }
    }
```

OUTPUT

	Input	Expected	Got	
	2 40 7	35 28 21 14 7	35 28 21 14 7	

Passed all tests!

QUOTIENT AND REMINDER

Write a C program to find the quotient and reminder of given integers.

For example:

Input	Result
12 3	4 0

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b,c;
   scanf("%d",&a);
   scanf("%d",&b);
   c=a/b;
   printf("%d \n",c);
   c=a%b;
   printf("%d",c);
}
```

OUTPUT

	Input	Expected	Got	
	12 3	4 0	4 0	

Passed all tests!

BIGGEST AMONG 3 INTEGERS

Write a C program to find the biggest among the given 3 integers?

For example:

Input	Result
10 20 30	30

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b,c;
   scanf("%d %d %d",&a,&b,&c);
   if(a>b&&a>c)
   {
      printf("%d",a);
   }
   else if (b>c&&c>a)
   {
      printf("%d",b);
   }
   else
   {
      printf("%d",c);
   }
}
```

OUTPUT

	Input	Expected	Got	
	10 20 30	30	30	

Passed all tests!

ODD OR EVEN

Write a C program to find whether the given integer is odd or even?

For example:

Input	Result
12	Even
11	Odd

PROGRAM

```
#include<stdio.h>
int main()
{
   int a;
   scanf("%d",&a);
   if(a%2)
   {
      printf("Odd");
   }
   else
   {
      printf("Even");
   }
}
```

OUTPUT

	Input	Expected	Got	
	12	Even	Even	
	11	Odd	Odd	

Passed all tests!

FACTORIAL OF N

Write a C program to find the factorial of given n.

For example:

Input	Result
5	120

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b=1,i;
   scanf("%d",&a);
   for(i=1;i<=a;i++)
   {
      b=b*i;
   }
      printf("%d",b);
}
```

OUTPUT

	Input	Expected	Got	
	5	120	120	

Passed all tests!

SUM OF FIRST N INTEGERS

Write a C program to find the sum first N natural numbers.

For example:

Input	Result
3	6

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b=0,i;
   scanf("%d",&a);
   for(i=1;i<=a;i++)
   {
      b+=i;
   }
   printf("%d",b);

}
```

OUTPUT

	Input	Expected	Got	
	3	6	6	

Passed all tests!

FIBONACCI SERIES

Write a C program to find the Nth term in the fibonacci series.

For example:

Input	Result
0	0
1	1
4	3

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b=0,c=1,sum=0,i;
   scanf("%d",&a);
   for(i=0;i<a-1;i++)
   {
   sum=b+c;
   b=c;
   c=sum;
   }
   if(a==1)
   {
   printf("1");
   }
   else
   {
      printf("%d",sum);
   }
}
```

OUTPUT

	Input	Expected	Got	
	0	0	0	
	1	1	1	
	4	3	3	

Passed all tests!

POWER OF INTEGERS

\
Write a C program to find the power of integers.

input:
a b
output:
a^b value

For example:

Input	Result
2 5	32

PROGRAM

```
#include<stdio.h>
#include<math.h>
int main()
{
   int a,b,c;
   scanf("%d %d",&a,&b);
   c=pow(a,b);
   printf("%d",c);
}
```

OUTPUT

	Input	Expected	Got	
	2 5	32	32	

Passed all tests!

PRIME OR NOT

Write a C program to find Whether the given integer is prime or not.

For example:

Input	Result
7	Prime
9	No Prime

PROGRAM

```
#include <stdio.h>
int main()
{
   int i,flag=1,a;
   scanf("%d",&a);
   for(i=2;i<a;i++)
   {
      if(a%i==0)
      {
         flag=1;
         break;
      }
      else
      {
         flag=0;
      }
   }
   if(flag==0)
   {
      printf("Prime");
   }
   else
   {
      printf("No Prime");
   }
}
```

OUTPUT

	Input	Expected	Got	
	7	Prime	Prime	
	9	No Prime	No Prime	

Passed all tests!

REVERSE OF GIVEN INTEGER

Write a C program to find the reverse of the given integer?

PROGRAM

```
#include<stdio.h>
int main()
{
   int a,b,c=0;
   scanf("%d",&a);
   while(a!=0)
   {
      b=a%10;
      c=c*10+b;
      a/=10;
   }
    printf("%d",c);
}
```

OUTPUT

	Input	Expected	Got	
	123	321	321	

Passed all tests!

FINDING TIME COMPLEXITY OF ALGORITHMS

PROGRAM 1

Problem 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)
        itt;
        s += i;
}
```

Note: No need of counter increment for declarations and scanfO and count variable printiO statements.

Input:
A positive Integer n

Output:
Print the value of the counter variable

For example:

Input	Result
9	12

PROGRAM

```
#include <stdio.h>
int main()
{
    int count=0;
    int n;
    scanf ("%d", &n);
    int i=1;
    count++;
    int s=1;
    count++;
    while(s<=n)
    {
        count++;
        i++;
```

```
                count++;
                s+=i;
                count++;
            }
            count++;
            printf("%d", count);
        }
```

OUTPUT

	Input	Expected	Got	
	9	12	12	
	4	9	9	

PROGRAM 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

PROGRAM

```
#include <stdio.h>
int main()
{
   int i,j,n;
   int count;
   count=0;
   scanf("%d",&n);
   if(n==1)
   {
      count++;
      printf("*");
   }
```

```
        else
        {
           count++;
           for(i=1;i<=n;i++)
           {
              count++;
              for(j=1;j<=n;j++)
              {
                 count++;
                 //printf("*");
                 count++;
                 //printf("*");
                 count++;
                 break;
                 count++;
              }count++;
           }count++;
        }
        printf("%d",count);
     }
```

OUTPUT

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

Passed all tests!

PROGRAM 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

PROGRAM

```
#include <stdio.h>
void factor(int num)
{
   int count=0;
   for(int i=1;i<=num;++i)
   {
      count++;
      if (num%i==0)
      {
         count++;
         //printf("%d",i);
      }
      count++;
   }
   count++;
   printf("%d",count);
}
int main()
{
   int num;
```

```
    scanf("%d",&num);
    factor(num);
}
```

OUTPUT

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

Passed all tests!

PROGRAM 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

PROGRAM

```
#include<stdio.h>
void function(int n)
{
   int c=0;
   c++;
   for(int i=n/2;i<n;i++)
   {
      c++;
     for(int j=1;j<n;j=2*j)
     {
        c++;
       for(int k=1;k<n;k=k*2)
        {
           c++;
           c++;
        }c++;
     }c++;
   }c++;
   printf("%d",c);
}
int main()
```

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

OUTPUT

	Input	Expected	Got	
	4	30	30	
	10	212	212	

Passed all tests!

PROGRAM 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

PROGRAM

```
#include<stdio.h>
void reverse(int n)
{
   int count=0;
   count++;
   int rev=0,remainder;
   count++;
   while(n!=0)
   {
      count++;
      remainder=n%10;
      count++;
      rev=rev*10+remainder;
      count++;
      n/=10;
      count++;
   }count++;
   //printf("rev");
   printf("%d",count);
```

```
}
int main()
{
   int n =0;
   scanf("%d",&n);
   reverse(n);
}
```

OUTPUT

	Input	Expected	Got	
	12	11	11	
	1234	19	19	

Passed all tests!

DIVIDE AND CONQUER

NUMBER OF ZEROS IN A GIVEN ARRAY

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.

PROGRAM

```
#include <stdio.h>
int count_zeros(int arr[], int left, int right) {
    if (left > right) {
        return 0;
    }
    if (left == right) {
        return arr[left] == 0 ? 1 : 0;
    }
    int mid = (left + right) / 2;
    return count_zeros(arr, left, mid) + count_zeros(arr, mid + 1, right);
}
int main() {
    int m;
    scanf("%d", &m);
    int arr[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }
    int number_of_zeros = count_zeros(arr, 0, m - 1);
    printf("%d", number_of_zeros);
    return 0;
}
```

OUTPUT

	Input	Expected	Got
	5 1 1	2	2

	Input	Expected	Got
	1 0 0		
	10 1 1 1 1 1 1 1 1 1 1	0	0
	8 0 0 0 0 0 0 0 0	8	8
	17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	2	2

	Input	Expected	Got
	0		

Passed all tests!

MAJORITY ELEMENT

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

PROGRAM

```
#include <stdio.h>
int majority(int* num, int numSize) {
    int candidate = 0;
    int count = 0;
    for (int i = 0; i < numSize; i++) {
        if (count == 0) {
            candidate = num[i];
        }
        count += (num[i] == candidate) ? 1 : -1;
    }
    return candidate;
}
int main() {
    int n;
```

```
    scanf("%d", &n);
    int num[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &num[i]);
    }
    int result = majority(num, n);
    printf("%d", result);
    return 0;
}
```

OUTPUT

	Input	Expected	Got	
	3 3 2 3	3	3	

Passed all tests!

FINDING FLOOR VALUE

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

PROGRAM

```
#include <stdio.h>
int findFloor(int array[], int n, int x) {
    int left = 0;
    int right = n - 1;
    int floorValue = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (array[mid] == x) {
            return array[mid];
        }
        else if (array[mid] < x) {
            floorValue = array[mid];
            left = mid + 1;
        }
        else {
            right = mid - 1;
        }
    }
    return floorValue;
}
int main() {
    int n;
    scanf("%d", &n);
    int array[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &array[i]);
    }
    int x;
```

```
    scanf("%d", &x);
    int floorValue = findFloor(array, n, x);
    if (floorValue != -1) {
        printf("%d", floorValue);
    } else {
        printf("%d", x);
    }
    return 0;
}
```

OUTPUT

	Input	Expected	Got	
	6 1 2 8 10 12 19 5	2	2	
	5 10 22 85 108 129 100	85	85	
	7 3 5 7 9 11 13 15 10	9	9	

Passed all tests!

TWO ELEMENTS SUM TO X

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”)

PROGRAM

```
#include <stdio.h>
int findPair(int array[], int left, int right, int x) {
   if (left >= right) {
      return 0;
   }
   if (array[left] + array[right] == x) {
      printf("%d\n%d\n", array[left], array[right]);
      return 1;
   }
   else if (array[left] + array[right] < x) {
      return findPair(array, left + 1, right, x);
   }
   else {
      return findPair(array, left, right - 1, x);
   }
}
int main() {
   int n;
   scanf("%d", &n);
```

```
    int array[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &array[i]);
    }
    int x;
    scanf("%d", &x);
    if (findPair(array, 0, n - 1, x)) {
    } else {
        printf("No\n");
    }
    return 0;
}
```

OUTPUT

	Input	Expected	Got	
	4 2 4 8 10 14	4 10	4 10	
	5 2 4 6 8 10 100	No	No	

Passed all tests!

IMPLEMENTATION OF QUICK SORT

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

PROGRAM

```
#include <stdio.h>
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition(int array[], int low, int high) {
    int pivot = array[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
            i++;
            swap(&array[i], &array[j]);
        }
    }
    swap(&array[i + 1], &array[high]);
    return i + 1;
}
void quickSort(int array[], int low, int high) {
    if (low < high) {
        int pi = partition(array, low, high);
        quickSort(array, low, pi - 1);
        quickSort(array, pi + 1, high);
    }
```

```
}
int main() {
    int n;
    scanf("%d", &n);
    int array[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &array[i]);
    }
    quickSort(array, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%d ", array[i]);
    }
    return 0;
}
```

OUTPUT

	Input	Expected	Got	
	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	
	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90 114	
	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	

Passed all tests!

GREEDYALGORITHMS

G-COIN PROBLEM

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.

PROGRAM

```
#include <stdio.h>
int Coins(int V) {
   int d[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
   int n = sizeof(d) / sizeof(d[0]);
   int coin = 0;
   for (int i = 0; i < n; i++) {
      while (V >= d[i]) {
         V -= d[i];
         coin++;
      }
   }

   return coin;
}
int main() {
   int V;
   scanf("%d", &V);
   int v = Coins(V);
   printf("%d\n",v);
   return 0;
}
```

OUTPUT

	Input	Expected	Got	
	49	5	5	

Passed all tests!

G-COOKIES PROBLEM

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

PROGRAM

```
#include <stdio.h>
void bubbleSort(int arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
int maxContentChildren(int g[], int gSize, int s[], int sSize) {
    bubbleSort(g, gSize);
    bubbleSort(s, sSize);
    int childIndex = 0;
    int cookieIndex = 0;
```

```
    while (childIndex < gSize && cookieIndex < sSize) {
        if (s[cookieIndex] >= g[childIndex]) {
            childIndex++;
        }
        cookieIndex++;
    }
    return childIndex;
}
int main() {
    int n;
    scanf("%d", &n);
    int g[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &g[i]);
    }
    int m;
    scanf("%d", &m);
    int s[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &s[i]);
    }
    int result = maxContentChildren( g,n, s, m);
    printf("%d\n",result);
    return 0;
}
```

OUTPUT

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

Passed all tests!

G-BURGER PROBLEM

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*

PROGRAM

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

```
        for (int i = 0; i < n-1; i++) {

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

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

OUTPUT

	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!

G-ARRAY SUM MAX PROBLEM

*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

PROGRAM

```
#include<stdio.h>
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++)
      {
         for (int j = i + 1; j < n; j++)
         {
            if (arr[i] > arr[j])
            {
               int temp =  arr[i];
               arr[i] = arr[j];
               arr[j] = temp;
            }

         }

      }
   int sum=0;
   for (int i = 0; i < n; i++) {
      sum+=arr[i]*i;
   }
```

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

OUTPUT

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!

G-PRODUCT OF ARRAY ELEMENTS-MINIMUM

*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*

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
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);
}
long long minSumOfProducts(int arrayOne[], int arrayTwo[], int n) {
   qsort(arrayOne, n, sizeof(int), compareAsc);
   qsort(arrayTwo, n, sizeof(int), compareDesc);

   long long minSum = 0;
   for (int i = 0; i < n; i++) {
      minSum += (long long)arrayOne[i] * arrayTwo[i];
   }
   return minSum;
```

```
}
int main() {
    int n;
    scanf("%d", &n);
    int arrayOne[n];
    int arrayTwo[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arrayOne[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &arrayTwo[i]);
    }
    long long result = minSumOfProducts(arrayOne, arrayTwo, n);
    printf("%lld\n", result);
    return 0;
}
```

OUTPUT

	Input	Expected	Got	
	3 1 2 3 4 5 6	28	28	
	4 7 5 1 2 1 3 4 1	22	22	

	Input	Expected	Got	
	5 20 10 30 10 40 8 9 4 3 10	590	590	

Passed all tests!

DYNAMIC PROGRAMMING

DP-PLAYING WITH NUMBERS

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

PROGRAM

```
#include <stdio.h>
long long countWays(int n) {
   long long dp[n + 1];
   dp[0] = 1;
   dp[1] = 1;
   dp[2] = 1;
   dp[3] = 2;
   for (int i =2; i <= n; i++) {
      dp[i] = dp[i - 1];
      if (i >= 3) {
         dp[i] += dp[i - 3];
      }
   }
```

```
    return dp[n];
}
int main() {
    int n;

    scanf("%d", &n);
    long long result = countWays(n);
    printf("%lld\n", result);
    return 0;
}
```

OUTPUT

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

Passed all tests!

DP-PLAYING WITH CHESSBOARD

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

PROGRAM

```
#include <stdio.h>
#define MAX 100
int maxMonetaryValue(int board[MAX][MAX], int n) {
   int dp[MAX][MAX];
   dp[0][0] = board[0][0];
   for (int j = 1; j < n; j++) {
```

```
            dp[0][j] = dp[0][j - 1] + board[0][j];
        }
        for (int i = 1; i < n; i++) {
            dp[i][0] = dp[i - 1][0] + board[i][0];
        }
        for (int i = 1; i < n; i++) {
            for (int j = 1; j < n; j++) {
                dp[i][j] = board[i][j] + (dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j -
1]);
            }
        }
        return dp[n - 1][n - 1];
    }
    int main() {
        int n;
        int board[MAX][MAX];

        scanf("%d", &n);
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                scanf("%d", &board[i][j]);
            }
        }
        int result = maxMonetaryValue(board, n);
        printf("%d\n", result);
        return 0;
    }
```

OUTPUT

	Input	Expected	Got	
	3 1 2 4 2 3 4 8 7 1	19	19	
	3	12	12	

	Input	Expected	Got	
	1 3 1 1 5 1 4 2 1			
	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	

Passed all tests!

DP-LONGEST COMMON SUBSEQUENCE

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*

PROGRAM

```
#include <stdio.h>
#include <string.h>
#define MAX 100 // Define maximum string length
int longestCommonSubsequence(char *s1, char *s2) {
    int n = strlen(s1);
    int m = strlen(s2);

    // Create a DP table
    int dp[MAX][MAX];
    // Initialize the DP table
    for (int i = 0; i <= n; i++) {
```

```
            for (int j = 0; j <= m; j++) {
                if (i == 0 || j == 0) {
                    dp[i][j] = 0; // Base case: empty string
                } else if (s1[i - 1] == s2[j - 1]) {
                    dp[i][j] = dp[i - 1][j - 1] + 1; // Characters match
                } else {
                    dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1]; // No match
                }
            }
        }
        return dp[n][m]; // Length of LCS
    }
    int main() {
        char s1[MAX], s2[MAX];
        // Input strings

        scanf("%s", s1);
        scanf("%s", s2);
        // Calculate and print the length of the longest common subsequence
        int result = longestCommonSubsequence(s1, s2);
        printf("%d\n", result);
        return 0;
    }
```

OUTPUT

	Input	Expected	Got	
	aab azb	2	2	
	ABCD ABCD	4	4	

Passed all tests!

DP-LONGEST NON-DECREASING SUBSEQUENCE

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

PROGRAM

```
#include <stdio.h>
#define MAX 100 // Define maximum size for the input sequence
int longestNonDecreasingSubsequence(int arr[], int n) {
   int dp[MAX] = {0}; // DP array to store lengths of LNDS
   // Initialize dp array
   for (int i = 0; i < n; i++) {
      dp[i] = 1; // Every element is at least a subsequence of length 1
   }
   // Fill the dp array
   for (int i = 1; i < n; i++) {
      for (int j = 0; j < i; j++) {
         if (arr[j] <= arr[i]) {
            dp[i] = (dp[i] > dp[j] + 1) ? dp[i] : (dp[j] + 1);
         }
      }
   }
   // Find the maximum length in dp array
   int maxLength = 0;
   for (int i = 0; i < n; i++) {
      if (dp[i] > maxLength) {
         maxLength = dp[i];
      }
   }
   return maxLength; // Return the maximum length found
}
int main() {
   int n;
   int arr[MAX];
   scanf("%d", &n);

   for (int i = 0; i < n; i++) {
```

```
        scanf("%d", &arr[i]);
    }
    // Calculate and print the length of the longest non-decreasing subsequence
    int result = longestNonDecreasingSubsequence(arr, n);
    printf("%d\n", result);
    return 0;
}
```

OUTPUT

	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!

COMPETITIVE PROGRAMMING

FINDING DUPLICATES-O(N^2) TIME COMPLEXITY,O(1) SPACE COMPLEXITY

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*

PROGRAM

```
#include <stdio.h>
int main() {
   int n, i, j, duplicate;

   scanf("%d", &n);

   int arr[n];

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

   for (i = 0; i < n; i++) {
      for (j = i + 1; j < n; j++) {
         if (arr[i] == arr[j]) {
            duplicate = arr[i];
            break;
         }
      }
      if (arr[i] == duplicate) {
         break;
      }
   }

   printf("%d\n", duplicate);
```

```
    return 0;
}
```

OUTPUT

	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!

FINDING DUPLICATES-O(N) TIME COMPLEXITY,O(1) SPACE COMPLEXITY

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*

PROGRAM

```
#include <stdio.h>
int main() {
   int n, i, j, duplicate;

   scanf("%d", &n);

   int arr[n];

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

   for (i = 0; i < n; i++) {
      for (j = i + 1; j < n; j++) {
         if (arr[i] == arr[j]) {
            duplicate = arr[i];
            break;
         }
      }
      if (arr[i] == duplicate) {
         break;
      }
   }

   printf("%d\n", duplicate);
```

```
        return 0;
    }
```

OUTPUT

	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!

*PRINT INTERSECTION OF 2 SORTED ARRAYS-O(M*N)TIME COMPLEXITY,O(1) SPACE COMPLEXITY*

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*

PROGRAM

```
#include <stdio.h>
int main() {
    int T;
    scanf("%d", &T);  // Number of test cases

    while (T--) {
        int N1, N2, i, j;
```

```
        // Read the size of the first array and the elements
        scanf("%d", &N1);
        int arr1[N1];
        for (i = 0; i < N1; i++) {
            scanf("%d", &arr1[i]);
        }
        // Read the size of the second array and the elements
        scanf("%d", &N2);
        int arr2[N2];
        for (i = 0; i < N2; i++) {
            scanf("%d", &arr2[i]);
        }
        // Initialize two pointers for both arrays
        i = 0;
        j = 0;
        // Find intersection of the two sorted arrays
        while (i < N1 && j < N2) {
            if (arr1[i] < arr2[j]) {
                i++;
            } else if (arr1[i] > arr2[j]) {
                j++;
            } else {
                // Print the common element
                printf("%d ", arr1[i]);
                i++;
                j++;
            }
        }
        printf("\n");  // Print a new line after each test case
    }
    return 0;
}
```

OUTPUT

	Input	*Expected*	*Got*	
	1 *3 10 17 57* *6* *2 7 10 15 57 246*	*10 57*	*10 57*	
	1 *6 1 2 3 4 5 6* *2* *1 6*	*1 6*	*1 6*	

Passed all tests!

PRINT INTERSECTION OF 2 SORTED ARRAYS-O(M+N)TIME COMPLEXITY,O(1) SPACE COMPLEXITY

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*

PROGRAM

```
#include <stdio.h>
int main() {
    int T;
    scanf("%d", &T);  // Number of test cases

    while (T--) {
        int N1, N2, i, j;
```

```
        // Read the size of the first array and the elements
        scanf("%d", &N1);
        int arr1[N1];
        for (i = 0; i < N1; i++) {
            scanf("%d", &arr1[i]);
        }
        // Read the size of the second array and the elements
        scanf("%d", &N2);
        int arr2[N2];
        for (i = 0; i < N2; i++) {
            scanf("%d", &arr2[i]);
        }
        // Initialize two pointers for both arrays
        i = 0;
        j = 0;
        // Find intersection of the two sorted arrays
        while (i < N1 && j < N2) {
            if (arr1[i] < arr2[j]) {
                i++;
            } else if (arr1[i] > arr2[j]) {
                j++;
            } else {
                // Print the common element
                printf("%d ", arr1[i]);
                i++;
                j++;
            }
        }
        printf("\n");  // Print a new line after each test case
    }
    return 0;
}
```

OUTPUT

	Input	*Expected*	*Got*	
	1 *3 10 17 57* *6* *2 7 10 15 57 246*	*10 57*	*10 57*	
	1 *6 1 2 3 4 5 6* *2* *1 6*	*1 6*	*1 6*	

Passed all tests!

PAIR WITH DIFFERENCE-O(N^2)TIME COMPLEXITY,O(1) SPACE COMPLEXITY

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*

PROGRAM

```
#include <stdio.h>
int main() {
    int n, k, i, j;

    // Input the number of elements in the array
    scanf("%d", &n);

    int arr[n];

    // Input the array elements
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Input the value of k
    scanf("%d", &k);

    // Use two pointers approach to find if the condition is met
    i = 0;
```

```
    j = 1;

    while (i < n && j < n) {
        if (i != j && arr[j] - arr[i] == k) {
            printf("1\n");
            return 0;
        } else if (arr[j] - arr[i] < k) {
            j++;  // Increase j to get a bigger difference
        } else {
            i++;  // Increase i to reduce the difference
        }
    }

    // If no such pair is found
    printf("0\n");
    return 0;
}
```

OUTPUT

	Input	Expected	Got	
	3 1 3 5 4	1	1	
	10 1 4 6 8 12 14 15 20 21 25 1	1	1	
	10 1 2 3 5 11 14 16 24 28 29 0	0	0	
	10 0 2 3 7 13 14 15 20 24 25 10	1	1	

Passed all tests!

PAIR WITH DIFFERENCE -O(N) TIME COMPLEXITY,O(1) SPACE COMPLEXITY

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*

PROGRAM

```
#include <stdio.h>
int main() {
   int n, k, i = 0, j = 1;

   // Input the number of elements in the array
   scanf("%d", &n);

   int arr[n];

   // Input the array elements
   for (i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   // Input the value of k
   scanf("%d", &k);

   // Reset i and j for the two-pointer approach
   i = 0;
```

```
        j = 1;

        // Two-pointer technique
        while (i < n && j < n) {
            if (i != j && arr[j] - arr[i] == k) {
                printf("1\n");
                return 0;
            } else if (arr[j] - arr[i] < k) {
                j++;  // Increase j to get a larger difference
            } else {
                i++;  // Increase i to reduce the difference
            }
        }

        // If no such pair is found
        printf("0\n");
        return 0;
    }
```

OUTPUT

	Input	Expected	Got	
	3 1 3 5 4	1	1	
	10 1 4 6 8 12 14 15 20 21 25 1	1	1	
	10 1 2 3 5 11 14 16 24 28 29 0	0	0	
	10 0 2 3 7 13 14 15 20 24 25 10	1	1	

Passed all tests!