LAB MANUAL

NAME : THARUN RAJ I

DEPARTMENT : CSE

SECTION : F

YEAR : 2ND

ROLL NUMBER : 230701362

SUBJECT : DESIGN AND ANALYSIS OF ALGORITHMS

EXPNO:1

DATE: 8/8/24

FINDING TIME COMPLEXITY OF ALGORITHMS

CounterMethod

Q1)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

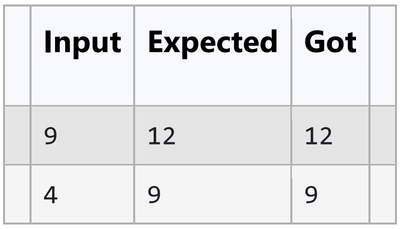
For example:

Input	Result
9	12

Code:

```
#include<stdio.h>
void function (int n)
{
   int c=0;
   int i= 1;c++;
   int s =1;c++;
   while(s <= n)
   {
       i++;c++;
       s += i;c++;
```

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



	Input	Expected	Got	
	9	12	12	
	4	9	9	

Q2) 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

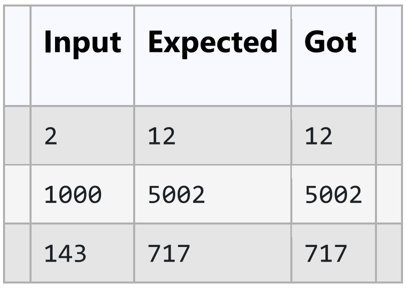
Output:

Print the value of the counter variable

Code:

```
#include<stdio.h>

void func(int n)

{

  int c=0;

  if(n==1)

  {

   c++;

   printf("*");

  }

  else

  { c++;

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

   { c++;

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

    {  c++;c++;c++;

      //printf("*");
```

```
        //printf("*");
        break;
      }c++;
    }c++;
  }
  printf("%d",c);
 }
int main(){
   int n;
   scanf("%d",&n);
   func(n);
}
```



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

Q3) 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

CODE:

```
#include<stdio.h>

void Factor(int num)

 {  int c=0;

   for (int i=1; i<=num;++i)

   { c++;

    if (num % i== 0)

      {c++;

        //printf("%d ", i);

      }c++;

   }

   c++;

   printf("%d",c);

  }

int main(){

   int n;

   scanf("%d",&n);

   Factor(n);

}
```

Input	Expected	Got		
	12	31	31	

Input	Expected	Got		
	25	54	54	
	4	12	12	🟩

Q4) Convert the following algorithm into a program and find its time complexity using counter method.

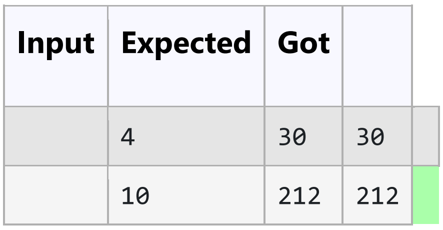
```
void function(int n)
{
    int c= 0;
    for(int i=n/2; i<n; i++)
        for(int j=1; j<n; j = 2 * j)
            for(int k=1; k<n; k = k * 2)
                c++;
}
```

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

CODE:

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



Input	Expected	Got		
	4	30	30	
	10	212	212	🟩

Q5) 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

CODE:

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

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

Input	Expected	Got		
	12	11	11	
	1234	19	19	🟩

Greedy Algorithms

ExpNo:2

Date:25/08/24

1G COIN PROBLEM

Q1) 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.

CODE:

```
#include<stdio.h>

int main(){

   int ar[10]={1,2,5,10,20,50,100,500,1000};

   int n,i=0,co=0,f=0;

   scanf("%d",&n);

   while(f!=1){

   do{

      i++;

    }while(ar[i]<=n);

    if(n%ar[i-1]!=0){

     co+=n/ar[i-1];

     n=n%ar[i-1];

     i=0;
    }
   else{
     co+=n/ar[i-1];
     f=1;
     }


   }
   printf("%d",co);



}
```

Input	Expected	Got	
	49	5	5

2G COOKIES PROBLEM

Q2) 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

CODE:

```
#include<stdio.h>

int main(){


  int n1,n2,p,temp,co=0;

  scanf("%d",&n1);

  int g[n1];

  for(int i=0;i<n1;i++){

    scanf("%d",&g[i]);
```

```
}
scanf("%d",&n2);
int s[n2];
for(int i=0;i<n2;i++){
   scanf("%d",&s[i]);
}
int max=s[0],k=1;
for(int i=0;i<n2;i++){
   for(int j=0;j<n2-i;j++){
      if(s[j]>=max){
        max=s[j];
        p=j;
      }
   }
   temp=s[n2-k];
   s[n2-k]=max;
   s[p]=temp;
   max=s[0];k++;
}k=1;max=g[0];
 for(int i=0;i<n1;i++){
   for(int j=0;j<n1-i;j++){
      if(g[j]>=max){
        max=s[j];
        p=j;
      }
   }
   temp=g[n1-k];
   g[n1-k]=max;
   g[p]=temp;
   max=g[0];k++;
}int j=0;
for(int i=0;i<n1;i++){
   while(j<n2){
   if(g[i]>=s[j]){
      co++;break;}
     j++;
     }
   }
   printf("%d",co);


}
```

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

3G BURGER PROBLEM

Q3) 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

CODE:

```
#include<stdio.h>
int powr(int b,int a){
  int ans=1;
  if(a==0)
   return 1;
  for(int j=0;j<a;j++){
    ans=ans*b;
  }
  return ans;
}
int main(){
  int f=0,n,temp,p,min,sum=0;
  scanf("%d",&n);
  int ar[n];
  for(int i=0;i<n;i++){
    scanf("%d",&ar[i]);
  }
  int end=n-1;
  for(int i=0;i<n;i++){
    min=100;f=0;
    for(int j=0;j<n-i;j++){
       if(min>ar[j]){
          min=ar[j];
          p=j;f=1;
       }
    }
   // printf("min%d pos%d ",min,p);
    if(f==1){
       temp=ar[end];
       ar[end]=ar[p];
       ar[p]=temp;
       end--;
    }
   /* for(int i=0;i<n;i++){
    printf("%d ",ar[i]);
    }
    printf("\n");*/

  }
  int b=n;
  for(int i=0;i<n;i++){
    sum=sum+(powr(b,i)*ar[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	

4G ARRAY SUM MAX PROBLEM

Q4) 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

CODE:

```
#include<stdio.h>
int main(){
  int f=0,n,temp,p,min,sum=0;
  scanf("%d",&n);
  int ar[n];
  for(int i=0;i<n;i++){
    scanf("%d",&ar[i]);
  }
  int end=n-1;
  for(int i=0;i<n;i++){
    min=100;f=0;
    for(int j=0;j<n-i;j++){
      if(min>ar[j]){
        min=ar[j];
        p=j;f=1;
      }
    }
    if(f==1){
      temp=ar[end];
      ar[end]=ar[p];
      ar[p]=temp;
      end--;
    }



  }
  p=0;
  for(int i=n-1;i>=0;i--){
   sum=sum+(i*ar[p]);
   p++;
  }
  printf("%d",sum);

}
```

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	🟩

5G PRODUCT OF ARRAY ELEMENTS -MIN

Q5) 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

CODE:

```
#include<stdio.h>

int main(){

   int f=0,n,temp,p,min,sum=0;

   scanf("%d",&n);

   int ar[n],ar1[n];

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

      scanf("%d",&ar[i]);

   }

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

      scanf("%d",&ar1[i]);

   }

   int end=n-1;

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

      min=100;f=0;

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

         if(min>ar[j]){

            min=ar[j];

            p=j;f=1;

         }
   }


   if(f==1){
      temp=ar[end];
      ar[end]=ar[p];
      ar[p]=temp;
      end--;
   }


}
end=n-1;
for(int i=0;i<n;i++){
   min=100;f=0;
   for(int j=0;j<n-i;j++){
      if(min>ar1[j]){
         min=ar1[j];
         p=j;f=1;
      }


   }


   if(f==1){
      temp=ar1[end];
      ar1[end]=ar1[p];
      ar1[p]=temp;
      end--;
   }


}
p=n-1;
for(int i=0;i<n;i++){
   sum=sum+(ar[i]*ar1[p]);
   p--;
```

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



}
```

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	

Divide and Conquer

ExpNo:3

Date:31/08/24

1-Number of Zeros in a Given Array

Q1) **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.

CODE:

```
#include<stdio.h>

void part(int ar[],int mid,int n){

   if((ar[mid]==0 && ar[mid-1]==1) ||(ar[mid]==0 && mid==0)){

      printf("%d",(n-1)-mid+1);

   }

   if(ar[mid]==1 && ar[mid+1]==0){

     printf("%d",(n-1)-mid);

   }

   if(ar[mid]==0 && ar[mid-1]==0)

      part(ar,mid/2,n);

   if(ar[mid]==1 && ar[mid+1]==1)

    part(ar,(mid+n)/2,n);

  }

int main(){

   int m;

   scanf("%d",&m);

   int ar[m];

   for(int i=0;i<m;i++){
      scanf("%d",&ar[i]);
    }
    if(ar[m-1]==1)
      printf("%d",0);
    else if(ar[0]==0)
      printf("%d",m);
    else{
    part(ar,m/2,m);
    }
}
```

Input	Expected	Got		
	5 1 1 1 0 0	2	2	
	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	2	2	🟩

Input	Expected	Got	
	1 1 1 1 1 0 0		

2-Majority Element

Q2)Given an array nums of size n, return *the majority element*.

The majority element is the element that appears more than $\lfloor n / 2 \rfloor$ 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

Code:

```
#include<stdio.h>

int main(){

  int elt,n,co=0,fco=0;

  scanf("%d",&n);
```

```
    int ar[n];
    for(int i=0;i<n;i++){
       scanf("%d",&ar[i]);
    }
   for(int i=0;i<n;i++){
      co=0;
      for(int j=0;j<n;j++){
         if(ar[j]==ar[i])
           co++;
      }
      if(fco<co){
         fco=co;
         elt=ar[i];
      }
   }
   printf("%d",elt);
}
```

Input	Expected	Got		
	3 3 2 3	3	3	

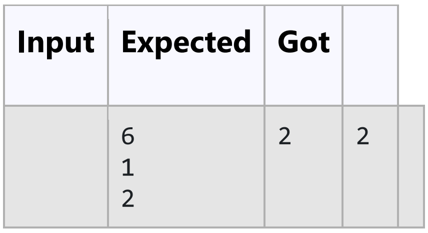
3-FINDING FLOOR VALUE

Q3) 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

CODE:

```
#include<stdio.h>

void find(int x,int mid,int b[]){

   if(b[mid]<x && b[mid+1]>=x)

      printf("%d",b[mid]);

   if(b[mid]>=x){

      find(x,mid/2,b);

   }

}

int main(){

 int n,x;

 scanf("%d",&n);

 int ar[n];

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

    scanf("%d",&ar[i]);

 }

 scanf("%d",&x);

 find(x,n/2,ar);

}
```



Input	Expected	Got		
	6 1 2	2	2	

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

4-Two Element Sum To X

Q4) 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**

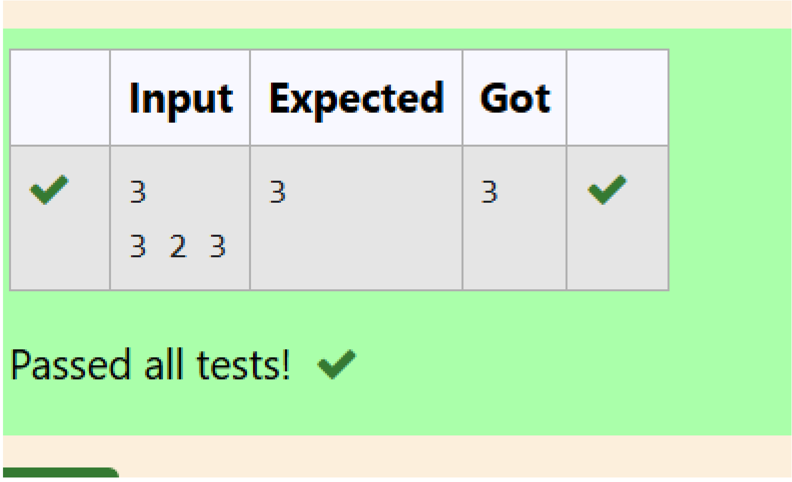
For example:

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

Code:

```
#include<stdio.h>

int main(){

   int elt,n,co=0,fco=0;

   scanf("%d",&n);

   int ar[n];
```

```
    for(int i=0;i<n;i++){
      scanf("%d",&ar[i]);
    }
   for(int i=0;i<n;i++){
     co=0;
     for(int j=0;j<n;j++){
       if(ar[j]==ar[i])
         co++;
     }
     if(fco<co){
       fco=co;
       elt=ar[i];
     }
   }
   printf("%d",elt);
}
```



	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

5-G-Product of Array elements-Minimum

Q5) 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.

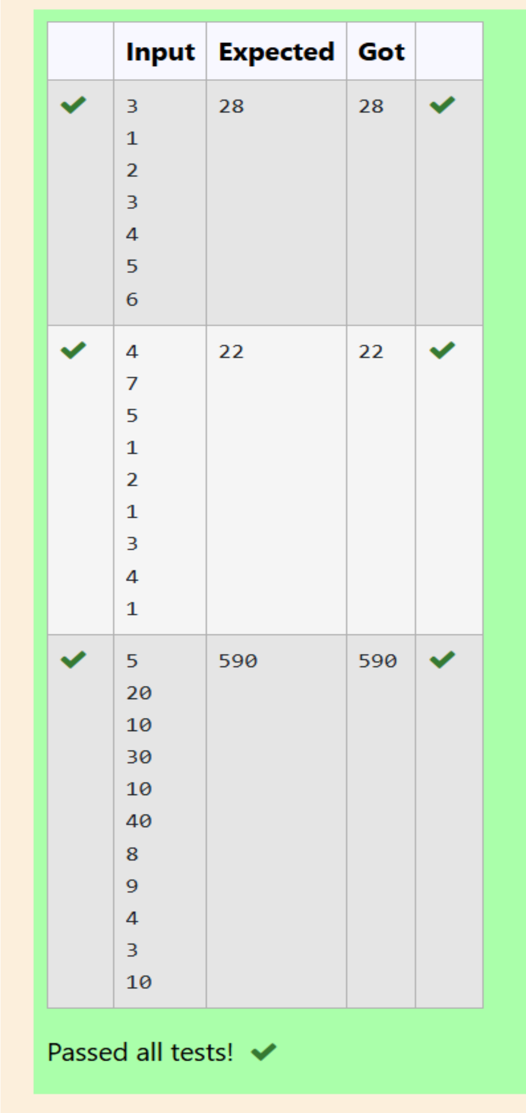
For example:

Input	Result
3 1 2 3 4 5 6	28

Code:

```
#include<stdio.h>
int main(){
  int f=0,n,temp,p,min,sum=0;
  scanf("%d",&n);
  int ar[n],ar1[n];
  for(int i=0;i<n;i++){
    scanf("%d",&ar[i]);
  }
  for(int i=0;i<n;i++){
    scanf("%d",&ar1[i]);
  }
  int end=n-1;
  for(int i=0;i<n;i++){
    min=100;f=0;
    for(int j=0;j<n-i;j++){
      if(min>ar[j]){
            min=ar[j];
            p=j;f=1;
        }

    }

    if(f==1){
        temp=ar[end];
        ar[end]=ar[p];
        ar[p]=temp;
        end--;
    }

}
end=n-1;
for(int i=0;i<n;i++){
    min=100;f=0;
    for(int j=0;j<n-i;j++){
        if(min>ar1[j]){
            min=ar1[j];
            p=j;f=1;
        }

    }

    if(f==1){
        temp=ar1[end];
        ar1[end]=ar1[p];
        ar1[p]=temp;
        end--;
    }

}
p=n-1;
```

```
    for(int i=0;i<n;i++){
       sum=sum+(ar[i]*ar1[p]);
       p--;


    }
    printf("%d",sum);




}
```



	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! ✓

DYNAMIC PROGRAMMING

ExpNo:4

Date:16/11/24

1-DP-Playing with Numbers

Q1) 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

Code:

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

```
int main(){
    int n;
    scanf("%d",&n);
    long sum=0;
    long a=1,b=1,c=2;
    if(n==1 || n==2)
      printf("%ld",a);
    else if(n==3)
       printf("%ld",c);
    else{
    for(int i=4;i<=n;i++){
        sum=a+c;
        a=b;
        b=c;
        c=sum;
    }
    printf("%ld",sum);
    }
}
```

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

2-DP-Playing with chessboard

Q2) 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

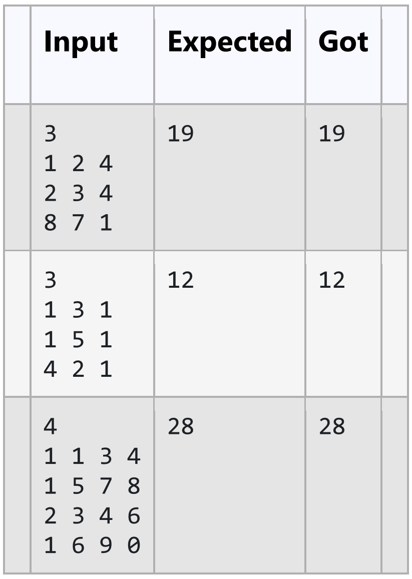
Output Format

Print Maximum monetary value of the path

Code:

```
#include<stdio.h>

int main(){

   int n,co,i,j,x,y;

   scanf("%d",&n);

   int ar[n][n];

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

      for(j=0;j<n;j++){

         scanf("%d",&ar[i][j]);

      }

   }

   i=0;j=0;co=ar[0][0];
```

```
while(1){


  if(i!=n-1 && j!=n-1){

    if(ar[i][j+1]>ar[i+1][j]){

       x=i;

       y=j+1;

       co=co+ar[i][j+1];


    }

    else if(ar[i][j+1]<ar[i+1][j]){

       x=i+1;

       y=j;

       co=co+ar[i+1][j];

    }

    else{

      x=i+1;

      y=j;

      co=co+ar[i+1][j];

    }

    i=x;

    j=y;


  }


  else{

     if(i==n-1){

        j=j+1;

        while(j<n){

           co=co+ar[i][j];

           j++;

        }

     }

     if(j==n-1){

        i=i+1;
        while(i<n){
          co=co+ar[i][j];
          i++;
        }
      }

      break;
    }

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



	Input	Expected	Got	
	3 1 2 4 2 3 4 8 7 1	19	19	
	3 1 3 1 1 5 1 4 2 1	12	12	
	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	

3-DP-Longest Common Subsequence

Q3) 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

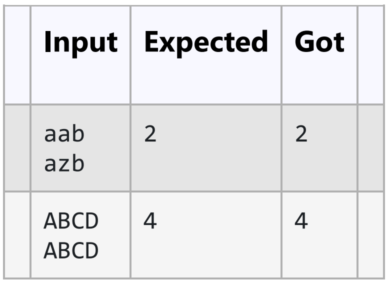
For example:

Input	Result
aab azb	2

Code:

```
#include<stdio.h>

#include<string.h>

int main(){

   char s1[10];

   char s2[10];

   int co=0,k,i,j,f;

   scanf("%s",s1);

   scanf("%s",s2);

   for(i=0;i<strlen(s1);i++){

      f=0;
```

```
        for(k=0;k<i;k++){
          if(s1[i]==s1[k] && i!=k){
            f=1;
          }

        }
        if(f==0){
          for(j=0;j<strlen(s2);j++){
            if(s1[i]==s2[j])
              co++;
          }
        }
    }
    printf("%d",co);
}
```



	Input	Expected	Got	
	aab azb	2	2	
	ABCD ABCD	4	4	

4-DP-Longest non-decreasing Subsequence

Q4) 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

Code:

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

  scanf("%d",&n);
   int ar[n];
  for(i=0;i<n;i++){
    scanf("%d",&ar[i]);
  }
  for(i=0;i<n;i++){
    co=1;
    for(j=i+1;j<n;j++){
      if(ar[j]>=ar[i] && ar[j-1]<=ar[j]){
       co++;
      }
      else{
       if(pco<co)
        pco=co;
       co=2;
      }
```

```
        }
        if(co>pco)
            pco=co;
    }
    printf("%d",pco);
}
```

	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	

COMPETITIVE PROGRAMMIG

ExpNo:5

Date:20/08/24

1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity

Q1) 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

CODE:

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

    for(int j=0;j<n;j++){

      if(ar[i]==ar[j]){

        co++;

        if(co>1)

          break;

      }

     }

    if(co>1){

     printf("%d",ar[i]);

     break;

    }

    co=0;

  }

}
```

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	

2-Finding Duplicates-O(n) Time Complexity,O(1) Space Complexity

Q2) 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

CODE:

```
#include<stdio.h>

int main(){

   int n;

   scanf("%d",&n);

   int ar[100]={0},x;

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

      scanf("%d",&x);

      if(ar[x%n]!=0 && ar[x%n]==x){

         printf("%d",x);

      }

      else{

         ar[x%n]=x;

      }

   }
}
```

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	🟩

3-Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity

Q3) 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	10 57

Input	Result

3 10 17 57
6
2 7 10 15 57 246	

Code:

```
#include<stdio.h>
int main(){
   int t,n1,n2,i,j,k;
   scanf("%d",&t);
   for(i=0;i<t;i++){
      scanf("%d",&n1);
      int ar1[n1];
      for(j=0;j<n1;j++){
         scanf("%d",&ar1[j]);
      }
      scanf("%d",&n2);
      int ar2[n2];
      for(j=0;j<n2;j++){
         scanf("%d",&ar2[j]);
      }
      for(j=0;j<n1;j++){
        // printf("j:%d\n",ar1[j]);
         for(k=0;k<n2;k++){
           // printf("k:%d\n",ar2[k]);
            if(ar1[j]==ar2[k]){
               printf("%d ",ar1[j]);
            }
         }
      }
   }
}
```

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	

4-Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity

Q4) 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	10 57

Input	Result

3 10 17 57
6
2 7 10 15 57 246	

Code:

```
#include<stdio.h>
int main(){
   int t,n1,n2,i,j,k,pk=0;
   scanf("%d",&t);
   for(i=0;i<t;i++){
      scanf("%d",&n1);
      int ar1[n1];
      for(j=0;j<n1;j++){
         scanf("%d",&ar1[j]);
      }
      scanf("%d",&n2);
      int ar2[n2];
      for(j=0;j<n2;j++){
         scanf("%d",&ar2[j]);
      }

      for(j=0;j<n1;j++){
         k=pk;
         while(k<n2){
            if(ar1[j]==ar2[k]){
               printf("%d ",ar1[j]);
            }
            if(ar2[k]>ar1[j]){
              pk=k;
              break;
            }
            k++;
```

```
        }
      }
    }
  }
```

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	🟩

5-Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity

Q5) 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

Code:

```
#include<stdio.h>

int main(){

   int n,i,k,f=0,j;

   scanf("%d",&n);

   int ar[n];

   for(i=0;i<n;i++){
        scanf("%d",&ar[i]);
    }
    scanf("%d",&k);
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(ar[j]-ar[i]==k){
              printf("1");f=1;}
        }
        if(f==1)
          break;
    }
    if(f==0)
      printf("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	
	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	

6-Pair with Difference -O(n) Time Complexity,O(1) Space Complexity

Q6) 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

Code:

```
#include<stdio.h>
int main(){
   int n,i,k,f=0,j;
   scanf("%d",&n);
   int ar[n];
   for(i=0;i<n;i++){
        scanf("%d",&ar[i]);
    }
    scanf("%d",&k);
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(ar[j]-ar[i]==k){
              printf("1");f=1;}
        }
        if(f==1)
          break;
    }
    if(f==0)
      printf("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	
	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	