```
/*Write a program to for finding one missing number and one duplicate in
an array of n integers where each number lies between 1 and n.
The program should handle additional constraints and provide an optimized
solution with minimal space complexity.

Detailed Problem Requirements:
Input Specifications:
Accept an array A of size n, where each element lies between 1 and n.
The array contains exactly one missing number and one duplicate number.
Constraints:
The array must not be modified.
No additional array or data structure should be used, but constant extra
space (apart from variables) is allowed.
Output Specifications:
Identify and display the missing number and the duplicate number.
If the input array violates constraints, display an appropriate error
message.

Algorithmic Constraints:
Use mathematical properties like the sum and sum of squares to find the
missing and duplicate numbers efficiently:
The sum of integers from 1 to n is S = (n * (n+1)) / 2
The sum of squares of integers from 1 to n is SS = (n * (n+1) * (2n+1)) /
6

Error Handling:
If the array has no missing or duplicate numbers, detect and report this
anomaly.
Validate that all array elements are integers within the range [1,n].
Output Requirements:
Display the original array and calculated values (e.g., sums and
differences).
Clearly show the duplicate and missing numbers.

Example Execution:
Input:
n=5n = 5n=5
Array: [1,2,2,4,5]
Processing:
Compute the expected sum and sum of squares:
```

```
S = (n * (n+1)) / 2= (5 * (6)) / 2=15
SS = (n * (n+1) * (2n+1)) / 6 = (5 * (6) * (11)) / 6=55
Calculate the actual sum and sum of squares:
S actual = 1+2+2+4+5 = 14
SS actual = 1^2 + 2^2 + 2^2 + 4^2 + 5^2 = 50
Derive the equations:
Missing - Duplicate = S -S actual = 15 - 14 = 1
Missing^2 - Duplicate^2 = SS - SS actual = 55-50=5
Solve the equations to find:
Missing = 3
Duplicate = 2
Output:
Original Array: [1, 2, 2, 4, 5]
Missing Number: 3
Duplicate Number: 2*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int array[20]={0};
int test1(int a[20],int n)
{
    int count =0;
    for(int i=0;i<n-1;i++)
    for(int j=i+1;j<n;j++)
    {
        if(a[i]==a[j])
        count++;
    }
    if(count==1)
    return 1;
    else
    return 0;

}
int test2(int a[20],int n)
{
    int flag = 1;
    for(int i=0;i<n;i++)
    if(a[i]>n)
    {
```

```
    flag=0;
    break;
    }
    if(flag==0)
    return 0;
    else
    return 1;

}
int test3(int a[20],int n)
{
    int flag=1;
    for(int i=0;i<n;i++)
    if(a[i]!=(i+1))
    {flag=0;
    break;
    }
    if(flag == 0)
    return 0;
    else
    return 1;
}
int Sum(int a[20],int n)
{
    int s=0;
    for(int i=0;i<n;i++)
    s+=a[i];
    return s;

}

int SSum(int a[20],int n)
{
    int ss=0;
    for(int i=0;i<n;i++)
    ss+=pow(a[i],2);
    return ss;

}
void print(int a[20],int n)
```

```
{
    for(int i=0;i<n;i++)
    printf("%d ",a[i]);
}
int dup(int a[20],int n)
{
    for(int i=0;i<n-1;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(a[i]==a[j])
            return a[i];
        }
    }

}
void main(){
    int n,sum=0,ssum=0,s,ss,mcount=0,missing,duplicate;
    int flag1, flag2,flag3;
    printf("Enter the size of array\n");
    scanf("%d",&n);
    printf("Enter the numbers\n");
    for(int i=0;i<n;i++)
    {
            scanf("%d",&array[i]);

    }
     flag2=test2(array,n);
    flag1=test1(array,n);
    if(flag1==0 ||flag2 ==0 || flag3 ==0)
    {
        if(flag1==0)
        printf("This array doesn't have a duplicate \n");
        if(flag2==0)
        printf("The element in the array is greater than %d\n",n);
        if(flag3 ==0)
        printf("This array doesnt have a missing element\n ");

    printf("This array does not satisfy the input conditions\n");
    exit(0);
```

```
    }
    sum =(n * (n+1)) / 2;
    ssum =(n * (n+1) * (2*n+1)) / 6;
    duplicate=dup(array,n);
    missing=sum-Sum(array,n)+duplicate;
    printf("The original array is\n");
    for(int i=0;i<n;i++)
    printf("%d ",array[i]);

    printf("\n");
    printf("Missing - %d\n",missing);
    printf("Duplicate - %d\n",duplicate);
    printf("Actual sum of squares is %d\n",SSum(array,n));
    printf("Estimated sum of squares is %d\n",ssum);
    printf("Actual sum is %d\n",Sum(array,n));
    printf("Estimated sum is %d\n",sum);
    printf("Difference between estimated sum and actual sum is
%d\n",sum-Sum(array,n));
    printf("Difference between estimated sum of squares and actual sum of
squares is %d\n",ssum-SSum(array,n));
    printf("Difference between missing and duplicate value is %d
\n",missing-duplicate);



}
/*
Additional Requirements:
Extend the program to handle edge cases, such as:
An array where all elements are the same except for the duplicate.-
An array where the missing number is n or 1. - In this case the input
would contain an integer greater than n therefore it would
not satisfy the condition that input should be between 1 and n
Provide a detailed report of the steps and computations, including checks
for valid inputs.
*/
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc 162.c -o 162
Enter the size of array
5
Enter the numbers
1 2 2 4 5
1 2 2 4 5
Missing - 3
Duplicate - 2
Actual sum of squares is 50
Estimated sum of squares is 55
Actual sum is 14
Estimated sum is 15
Difference between estimated sum and actual sum is 1
Difference between estimated sum of squares and actual sum of squares is 5
Difference between missing and duplicate value is 1
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc 162.c -o 162
Enter the size of array
5
Enter the numbers
1 2 3 3 5
The original array is
1 2 3 3 5
Missing - 4
Duplicate - 3
Actual sum of squares is 48
Estimated sum of squares is 55
Actual sum is 14
Estimated sum is 15
Difference between estimated sum and actual sum is 1
Difference between estimated sum of squares and actual sum of squares is 7
Difference between missing and duplicate value is 1
PS D:\projects\quest\C>
```