Find the missing and repeating number

BRUTE FORCE :

Run loop from 1 to n as the range given in ques is 1 to n then using linear search find whether numbers 1 to n exist in array if it does then increment count as many times it occurs. If cnt is 2 then set repeating and cnt is 0 means missing.

```
#include <bits/stdc++.h>

pair<int,int> missingAndRepeating(vector<int> &arr, int n)
{
  int repeating=-1,missing=-1;
  for(int i=1;i<=n;i++)
  {
    int cnt=0;
    for(int j=0;j<arr.size();j++)
    {
      if(arr[j]==i)
      {
        cnt++;
      }
    }
    if(cnt==2)
      repeating=i;
    else if(cnt==0)
      missing=i;
    if(repeating!=-1 && missing!=-1)
      break;
  }
  return {missing,repeating};
}
```

- Time Complexity : O(N^2)
- Space Complexity : O(N)

Approach 2 : Hashing

```
#include <bits/stdc++.h>
```

```
pair<int,int> missingAndRepeating(vector<int> &arr, int n)
{
  // Write your code here
  int freq[n+1]={0};
  int repeating=-1,missing=-1;
  for(int i=0;i<arr.size();i++)
  {
    freq[arr[i]]++;
  }
  for(int j=1;j<=n;j++)
  {
    if(freq[j]==0)
      missing=j;
    else if (freq[j]==2)
      repeating=j;

    if(repeating !=-1 && missing !=-1)
      break;

  }
  return {missing,repeating};
}
```

Approach 3 : Using maths

summing up given numbers(S) and summing up 1 to n (Sn)

Let X be repeating number and Y be misisng

ex : 1 2 4 2

actual numbers = 1 2 3 4

S-Sn= 2-3

where 2 is repeating and 3 is missing so it becomes X-Y.

Similarly for sum of squares of the numbers it will be X^2 - Y^2 which gives (X+Y)*(X-Y)

substituting values it becomes (X-Y)=val1

Solve above two equation to get x and y.

Take care of range in this ques use long long

```
#include <bits/stdc++.h>

pair<int,int> missingAndRepeating(vector<int> &arr, int n1)
{
  // Write your code here
```

```
  long long n=arr.size();
  long long sum= (n*(n+1))/2;
  long long sumsq=(n*(n+1)*(2*n+1))/6;
  long long sn=0,sqn=0;
  for(int i=0;i<arr.size();i++)
  {
    sn+=arr[i];
    sqn+=((long long)arr[i]*(long long)arr[i]);
  }
  long long val1=sn-sum;
  long long val2=(sqn-sumsq)/val1;
  long long x=(val1+val2)/2;
  long long y=x-val1;
  return {(int)y,(int)x};
}
```

- Time complexity : O(N)
- Space Complexity : O(1).

Approach 4 : XOR operation

Only for knowledge

Xor of any two same number is always zero that is they cancel each other so we can use this to our benefit. We will XOR the given numbers and the numbers that were meant to be right that is 1 to N after xoring all the numbers only two numbers will be odd (one which is missing will occur 1 time and the other that is repeated will occur 3 times) rest all the numbers will be even occuring and cancel each other.

So to carry out this in programming we first take a variable xr which xor’s all elements the result of this operation will be an integer and we need to find the right most set bet in this result to find out our numbers after finding the bit we separate all the numbers into whether the bit that is set to 0 or one. Now we get one number for one and for zero.

Search zero element in the array and count the element freq if 2 then zero elem is repeating else one is repeating.

```
#include <bits/stdc++.h>

pair<int,int> missingAndRepeating(vector<int> &a, int num)
{
  int n = a.size(); // size of the array
```

```
    int xr = 0;

    //Step 1: Find XOR of all elements:
    for (int i = 0; i < n; i++) {
        xr = xr ^ a[i];
        xr = xr ^ (i + 1);
    }

    //Step 2: Find the differentiating bit number:
    int number = (xr & ~(xr - 1));

    //Step 3: Group the numbers:
    int zero = 0;
    int one = 0;
    for (int i = 0; i < n; i++) {
        //part of 1 group:
        if ((a[i] & number) != 0) {
            one = one ^ a[i];
        }
        //part of 0 group:
        else {
            zero = zero ^ a[i];
        }
    }

    for (int i = 1; i <= n; i++) {
        //part of 1 group:
        if ((i & number) != 0) {
            one = one ^ i;
        }
        //part of 0 group:
        else {
            zero = zero ^ i;
        }
    }

    // Last step: Identify the numbers:
    int cnt = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] == zero) cnt++;
    }

    if (cnt == 2) return {one, zero};
    return {zero, one};
}
```