EASY :

1. Find Smallest number which have n zero

```
class Solution
{
   public:
   bool solve(int num,int n)
   {
      int f=5;
      int c=0;
      while(f<=num)
      {
         c+=num/f;
         f*=5;
      }
      return c>=n;
   }
      int findNum(int n)
      {
      if(n==1)
      return 5;
      int l=0;
      int r=5*n;
     while(l<r)
     {
         int mid=(l+r)/2;
         if(solve(mid,n))
         r=mid;
         else
         l=mid+1;
      }
      return l;
   }
};
```

2. Product array puzzle

```
class Solution{
  public:
   vector<long long int> productExceptSelf(vector<long long int>& nums, int n)
   {
      long long int multi=1;
      vector<long long int>ans;
      for(int i=0;i<n;i++)
      {
```

```
            multi=1;
            for(int j=0;j<n;j++)
            if(j!=i)
            multi*=nums[j];
            ans.push_back(multi);
        }
        return ans;
    }
};
```

MEDIUM :

1. Missing number in AP

```
class Solution{
public:
    int inSequence(int A, int B, int C){
         if(C==0)
         {
           if(A==B)
           return 1;
           else
           return 0;
         }

       int n=1+(B-A)/C;
       return ((B-A)%C==0 && n>0);
    }
};
```

2. Kth element of 2 sorted arrays

```
class Solution{
    public:
    int kthElement(int arr1[], int arr2[], int n, int m, int k)
    {
        int i=0;
        int j=0;
        int c=0;
        int f=0;

        while(c!=k)
        {
            f=0;
            if(i<n && arr1[i]<arr2[j])
            i++;
```

```
        else if(j<m)
        {
          j++;
          f=1;
        }
        c++;
      }
      if(f==0)
      return arr1[i-1];
      return arr2[j-1];
   }
};
```

3. Minimum swaps to sort

```
class Solution
{
   public:
       int minSwaps(vector<int>&nums)
       {
          vector<int>v;
          v=nums;
          map<int,int>mp;
          int c=0;

          sort(v.begin(),v.end());

          for(int i=0;i<v.size();i++)
          mp[v[i]]=i;


          int it=0;
          while(it<nums.size())
          {
             if(v[it]!=nums[it])
             {
                swap(nums[mp[nums[it]]], nums[it]);
                c++;
             }
             else
             it++;
          }
          return c;
       }
};
```

4. Zero sum subarrays

```
class Solution{
   public:
   //Function to count subarrays with sum equal to 0.
   ll findSubarray(vector<ll> arr, int n )
   {
      vector<long long int>a(n+1);
      a[0]=0;
      for(int i=0;i<arr.size();i++)
      a[i+1]=a[i]+arr[i];
      unordered_map<long long int,long long int>mp;
      long long int ans=0;
      for(int i=0;i<=n;i++)
      {
         mp[a[i]]++;

         ans+=(mp[a[i]]-1);
      }
      return ans;

   }
};
```

5. Count triplet sum smaller than given sum

```
class Solution{
public:
       long long countTriplets(long long nums[], int n, long long sum)
       {
          sort(nums, nums+n);
          long long int ans= 0;
      for(int i=0;i<n-2;i++)
      {
        int j=i+1;
        int k=n-1;
        while(j<k)
        {
            if(nums[i]+nums[j]+nums[k]<sum)
            ans+=k-j, j++;
            else
            k--;
       }
   }
   return ans;
   }};
```

6. Stickler thief

```
class Solution
{
   public:
   int FindMaxSum(int nums[], int n)
   {
       vector<int>dp(n,0);
      if(n==1)
      return nums[0];
      else if(n==2)
      return max(nums[0],nums[1]);

      dp[0]=nums[0];
      dp[1]=max(nums[0],nums[1]);


      for(int i=2;i<n;i++)
      dp[i]=max(dp[i-1],dp[i-2]+nums[i]);



      return dp[n-1];

   }
};
```

7. Find all fours sum which is equal given sum

```
class Solution{
   public:
   vector<vector<int> > fourSum(vector<int> &nums, int target) {

      vector<vector<int>> ans;
      sort(nums.begin(), nums.end());

      for(int i=0; i<nums.size(); i++)
      {
         if (i != 0 && nums[i] == nums[i-1])
            continue;

          for(int j=i+1; j<nums.size(); j++)
          {
            if (j != i+1 && nums[j] == nums[j-1])
                continue;
```

```
            int target_temp= target - nums[i] - nums[j];
            int x = j+1;
             int y = nums.size()-1;
            while(x < y)
            {
               int tmp = nums[x] + nums[y];
               if (x != j+1 && nums[x] == nums[x-1])
               {
                  x++;
                  continue;
               }

               if(tmp > target_temp)
               y--;

               else if(tmp < target_temp)
               x++;

               else
               {
                  ans.push_back({nums[i], nums[j], nums[x], nums[y]});
                  x++;
                  y--;
               }
            }
         }
      }
      return ans;
   }
};
```

HARD :

1. Allocate minimum number of pages

```
bool is_pos(int a[], int n, int m, int ans)
   {
        int sum=0;
        int c=1;
        for(int i=0;i<n;i++)
        {
           if((sum+a[i])>ans)
           {
              c++;
              sum=a[i];
```

```
            }
            else
            sum+=a[i];
          }
          return (c<=m);
    }


class Solution
{
    public:
      int findPages(int A[], int N, int M)
    {
        int sum=0;
        int maxx=0;
     for(int i=0;i<N;i++)
     {
         sum+=A[i];
         maxx=max(maxx,A[i]);
     }
     int l=maxx;
     int r=sum;
     int res=0;
     while(l<=r)
     {
         int mid=(l+r)/2;

         if(is_pos(A,N,M,mid))
         {
            res=mid;
            r=mid-1;
         }
         else
         l=mid+1;
     }
     return res;
    }
};
```