Q) **https://www.codechef.com/problems/MAXSLAM**

C++ code:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
   int x;
   cin >> x;
   int year = 0;
   while ((x + 4 * year) < 25) {
   year++;
   }
   cout<<year;
}
```

Java code:

```
import java.util.*;
import java.lang.*;
import java.io.*;

class Codechef
{
            public static void main (String[] args) throws java.lang.Exception
            {
               Scanner sc =new Scanner(System.in);
             int x;
   x=sc.nextInt();
   int year = 0;
   while ((x + 4 * year) < 25) {
   year++;
   }
   System.out.println(year);
```

```
        }
}
```

Q) https://www.codechef.com/problems/RED23

C++ CODE:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
        int t;
        cin>>t;
        while(t>0){
          int x;
          cin>>x;
            while(x>1){
            if(x%2==0){
              x=x/2;
            }
            else if(x>3){
              x=x-3;
            }
            else{
              break;
            }

          }
           cout<<x<<endl;;
          t--;
        }
}
```

JAVA CODE:

```
import java.util.*;
import java.io.*;
class Codechef
{
        public static void main (String[] args)
        {
          Scanner sc=new Scanner(System.in);
int t;
        t=sc.nextInt();
        while(t>0){
          int x;
          x=sc.nextInt();
            while(x>1){
            if(x%2==0){
              x=x/2;
            }
            else if(x>3){
              x=x-3;
            }
            else{
              break;
            }

          }
           System.out.println(x);
          t--;
        }

        }
}
```

Q) https://www.codechef.com/problems/LTALL

C++ CODE:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
int t;
cin>>t;
while(t>0){
   int n;
   cin>>n;
  string input;
  cin>>input;
  int arr[n];
   for(int i=0;i<n;i++){
      arr[i]=input[i]-'0';
   }
   int i=0;
   while(i<n){
      if(arr[i]==1){
         if((i-1>=0) && (arr[i-1]==0)){
            arr[i-1]=1;
         }
         else if((i+1<=n-1) && (arr[i+1]==0)){
            arr[i+1]=1;
            i++;
         }
      }
      i++;
   }
   string ans="Yes";
```

```
   for(auto ele:arr){
      if(ele!=1){
         ans="No";
         break;
      }
   }
   cout<<ans<<endl;
   t--;
}
}
```

JAVA CODE:

```
import java.util.*;
import java.lang.*;
import java.io.*;

class Codechef
{
            public static void main (String[] args) throws java.lang.Exception
            {
               Scanner sc=new Scanner(System.in);
            int t;
t=sc.nextInt();
while(t>0){
   int n;
   n=sc.nextInt();
  String input;
  input=sc.next();
  int arr[]=new int[n];
   for(int i=0;i<n;i++){
      arr[i]=input.charAt(i)-'0';
```

```
    }
    int i=0;
    while(i<n){
        if(arr[i]==1){
            if((i-1>=0) && (arr[i-1]==0)){
                arr[i-1]=1;
            }
            else if((i+1<=n-1) && (arr[i+1]==0)){
                arr[i+1]=1;
                i++;
            }
        }
        i++;
    }
    String ans="Yes";
    for(int ele:arr){
        if(ele!=1){
            ans="No";
            break;
        }
    }
    System.out.println(ans);
    t--;
}
            }
}
```

Summary:

Container	Syntax	Returns
`string`	`str.find("sub")`	Index (`size_t`)
`vector`	`find(v.begin(), v.end(), val)`	Iterator
`set` / `map`	`s.find(val)` / `m.find(key)`	Iterator

Saved memory full

🔍 `find` and `erase` Summary in C++

Container	Find Syntax	Erase Syntax	Returns (Find)
`string`	`str.find("sub")`	`str.erase(pos, len)`	`size_t` (index)
`vector`	`find(v.begin(), v.end(), val)`	`v.erase(iterator)`	`iterator`
`set`	`s.find(val)`	`s.erase(iterator)` or `s.erase(val)`	`iterator`
`map`	`m.find(key)`	`m.erase(iterator)` or `m.erase(key)`	`iterator`
`unordered_set`	`us.find(val)`	`us.erase(iterator)` or `us.erase(val)`	`iterator`
`unordered_map`	`um.find(key)`	`um.erase(iterator)` or `um.erase(key)`	`iterator`
`list`	`find(lst.begin(), lst.end(), val)`	`lst.erase(iterator)`	`iterator`
`deque`	`find(dq.begin(), dq.end(), val)`	`dq.erase(iterator)`	`iterator`

maxA = *max_element(A.begin() + i + 1, A.end());

🔷 **insert at beginning**

v.insert(v.begin(),5);

- Inserts 5 at the **beginning**.
- Now: [5, 15]

v.push_back(10);

cout<<v.size()<<" "<<v.capacity();

⍰ push_back(10) adds 10 to the vector.

⍰ size() = number of elements → 1

⍰ capacity() = total memory allocated (implementation dependent). Usually, it's ≥ size and increases as needed.

🔷 clear()

v.clear();

- Removes all elements from the vector.
- v.size() becomes 0.

🔷 Descending sort

sort(v.begin(), v.end(), greater<int>());

int n= sizeof(arr)/sizeof(arr[0]);

Prime numbers within range

```
#include <iostream>

using namespace std;

int main() {
  int s=1;
  int e=5;

  int count_prime_num=0;

  for(int i=s;i<=e;i++){
    int count=0;
    for(int j=1;j<=i;j++){
      if(i%j==0){
        count++;
```

```
        }
      }
      if(count==2){
        count_prime_num++;
        cout<<i<<" ";
      }
    }
    cout<<endl;
    cout<<count_prime_num;

}
```

Reverse number sum between range

```
#include <iostream>

using namespace std;

int main() {
   int s=21;
    int e= 25;
    int sum=0;
    for(int i=s;i<=e;i++){
      int rev=0;
      int num=i;
      while(num>0){
        int rem=num%10;
        rev= rev*10+rem;
        num=num/10;
      }
      sum+=rev;
    }
```

```
    cout<<sum;
}
```

Two sum:

```
class Solution {
public:
  vector<int> twoSum(vector<int>& nums, int target) {
    vector<int> ans;
    unordered_map<int,int> mp;
    int n = nums.size();
    for(int i=0;i<n;i++){
      int remsum = target- nums[i];
      if(mp.find(remsum)!=mp.end()){
        ans.push_back(i);
        ans.push_back(mp[remsum]);
       break;
      }
      mp[nums[i]]=i;
    }
    return ans;
  }
};
```

Triplet Sum:

Brute force:

```
for(int i=0;i<n-2;i++){
    for(int j=i+1;j<n-1;j++){
      for(int k=j+1;k<n;k++){
        int sum= arr[i]+arr[j]+arr[k];
        cout<<sum<<" ";
      }
```

```
    }
  }
```

Better:

```
bool hasTripletSum(vector<int> &arr, int target) {
    int n=arr.size();
   for(int i=0;i<n;i++){
    unordered_set<int> st;
    for(int j=i+1;j<n;j++){
      int rem=target - arr[i] -arr[j];
        if(st.find(rem)!=st.end()){
          return true;
        }
        st.insert(arr[j]);
      }
  }
  return false;
  }
```

Optimal:

```
bool hasTripletSum(vector<int> &arr, int target) {
  int n = arr.size();
  sort(arr.begin(), arr.end());
  for (int i = 0; i < n; i++) {
    int l = i + 1, r = n - 1;
    int requiredSum = target - arr[i];
    while(l < r) {
      if(arr[l] + arr[r] == requiredSum)
        return true;
      if(arr[l] + arr[r] < requiredSum)
        l++;
```

```
        else if(arr[l] + arr[r] > requiredSum)
            r--;
      }
    }
    return false;
    }
```

Triplet sum (LEETCODE):

```
vector<vector<int>> threeSum(vector<int>& arr) {
      int n = arr.size();
    vector<vector<int>> ans;
    sort(arr.begin(), arr.k());
    for (int i = 0; i < n - 2; i++) {
      // Skip duplicates for the first element
      if (i != 0 && arr[i] == arr[i - 1]) continue;
      // Initialize two pointers:
      int j = i + 1;
      int k = n - 1;
      while (j < k) {
        int sum = arr[i] + arr[j] + arr[k];  // Calculate the sum of the triplet
        if (sum < 0) {
          j++;  // If sum is less than 0, move the `j` pointer to the right
        }
        else if (sum > 0) {
          k--;   // If sum is greater than 0, move the `k` pointer to the left
        }
        else {
          // If the sum equals 0, we found a valid triplet
          vector<int> temp = {arr[i], arr[j], arr[k]};
          ans.push_back(temp);  // Add the triplet to the answer list
```

```
            j++;  // Move the `j` pointer to the right
            k--;   // Move the `k` pointer to the left

            // Skip duplicates for `j`
            while (j < k && arr[j] == arr[j - 1]) j++;

            // Skip duplicates for `k`
            while (j < k && arr[k] == arr[k + 1]) k--;
        }
    }
  }
  return ans;  // Return the result containing all the triplets
  }
```

SUBARRAY 2 LOOPS:

```
void printAllSubarrays(vector<int>& arr) {
  int n = arr.size();

  for (int i = 0; i < n; ++i) {
    vector<int> subarray;
    for (int j = i; j < n; ++j) {
      subarray.push_back(arr[j]);  // extend subarray
      // Print current subarray
      for (int x : subarray) {
        cout << x << " ";
      }
    }
  }
}
```

☑ **Given:**

```
vector<vector<int>> vec;
```

◇ **1. Number of rows:**

```
int rows = vec.size();
```

◇ **2. Number of columns in a specific row (e.g., row 0):**

```
int cols = vec[0].size();
```

```
int arr[4][4]={{1,2,3,4},{5,6,7,8},{4,8,9,8}};

int row= sizeof(arr)/sizeof(arr[0]);
int col= sizeof(arr[0])/sizeof(arr[0][0]);
```

`vector<vector<int>> transpose(cols, vector<int>(rows));`

<u>Transpose of matrix:</u>

```
int rows = matrix.size();        // R = 2
int cols = matrix[0].size();      // C = 3

// Transposed matrix (C x R)
vector<vector<int>> transpose(cols, vector<int>(rows));

// Fill the transpose matrix
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        transpose[j][i] = matrix[i][j];
    }
}
```

<u>in-place transpose only works for square matrices.</u>

<u>Inplace one →</u>

```
for(int i=0; i<row; i++){
    for(int j=i+1; j<row; j++){
        swap(arr[i][j], arr[j][i]);
    }
}
```

- **Why use j = i + 1 in transpose?**

In a square matrix transpose, you want to **swap only the elements above (or below) the main diagonal** to avoid:

1. **Swapping back the same values** again.
2. **Touching the diagonal**, which doesn't need to be swapped.

- **First loop (Main or left diagonal: i == j):**

```
for(int i=0;i<row;i++){
    for(int j=0;j<col;j++){
        if(i==j){
            cout<<arr[i][j]<<" ";
        }
    }
}
```

- **Second loop (Anti-diagonal without center: i + j == row - 1 and i != j):**

```
for(int i=0;i<row;i++){
    for(int j=0;j<col;j++){
        if(i+j == row - 1 && i != j){
            cout<<arr[i][j]<<" ";
        }
    }
}
```

- **i == j → Main diagonal**
- **i + j == row - 1 → Anti-diagonal**

BOUNDARY ELEMENTS OF 2D MATRIX

```
for(int i = 0; i < row; i++) {
   for(int j = 0; j < col; j++) {
      if(i == 0 || j == 0 || i == row - 1 || j == col - 1) {
         cout << arr[i][j] << " ";
      }
   }
}
```

☑ **What this condition does:**

i == 0 → top row

j == 0 → first column

i == row - 1 → bottom row

j == col - 1 → last column

Together, this condition prints all the border elements of the matrix.

☑ **Z-Pattern Definition:**

For a square matrix like:

CopyEdit

1 2 3

4 5 6

7 8 9

The **Z-pattern** would be:

- **Top row** → 1 2 3
- **Diagonal from top-right to bottom-left** → 5
- **Bottom row** → 7 8 9

CODE:

```
int main() {
   int arr[3][3] = {
      {1, 2, 3},
      {4, 5, 6},
      {7, 8, 9}
   };

   int n = 3;  // since it's square

   // Top row
   for(int j = 0; j < n; j++) {
      cout << arr[0][j] << " ";
   }

   // Diagonal from top-right to bottom-left
   for(int i = 1; i < n - 1; i++) {
      cout << arr[i][n - i - 1] << " ";
   } // This skips i = 0 (top row) and i = n - 1 (bottom row).


   // Bottom row
   for(int j = 0; j < n; j++) {
      cout << arr[n - 1][j] << " ";
   }

   return 0;
}
```