Answer to the question no-01

1st iteration:

1st step: **7 2** 13 2 11 4 -> 2 7 13 2 11 4
2nd step: 2 **7 13** 2 11 4 -> 2 7 13 2 11 4
3rd step: 2 7 **13 2** 11 4 -> 2 7 2 13 11 4
4th step: 2 7 2 **13 11** 4 -> 2 7 2 11 13 4
5th step: 2 7 2 11 **13 4 ->** 2 7 2 11 4 13

2nd iteration:

1st step: **2 7** 2 11 4 13 -> 2 7 2 11 4 13
2nd step: 2 **7 2** 11 4 13 ->2 2 7 11 4 13
3rd step: 2 2 **7 11** 4 13 -> 2 2 7 11 4 13
4th step: 2 2 7 **11 4** 13 -> 2 2 7 4 11 13
5th step: 2 2 7 4 **11 13** -> 2 2 7 4 11 13

3rd iteration:

1st step: **2 2** 7 4 11 13 -> 2 2 7 4 11 13
2nd step: 2 **2 7** 4 11 13 -> 2 2 7 4 11 13
3rd step: 2 2 **7 4** 11 13 -> 2 2 4 7 11 13
4th step: 2 2 4 **7 11** 13 -> 2 2 4 7 11 13
5th step: 2 2 4 7 **11 13** -> 2 2 4 7 11 13

Answer to the question no-02

Array	Vector
1. We have to declare the size of the array first.	1. We do not need to declare the size of the vector.
2. We can not sort the array with built-in sort function	2. It is easy to sort the vector with Sort function.

Answer to the question no-03

```
for(int i=1;i<=n;i++)
{
        if(builtin_popcount(i) == 2)
        {
                for(int j=1;j<=n;j++)
                        cout<<i<<j<<endl;
        }
}
```

Note: builtin_popcount(i) returns the number of set bits in 'i'.
For example builtin_popcount(5) = 2. Because, 5 = $(101)_2$. So there are 2 set bits in 5.

Here the example means that if a decimal number's binary value starts with 1 and ends with 1, it means that the value has 2 set bits.
We can get that number by using this formula-> 2^n+1 ,here n=1,2,3…
Suppose the 2nd loop will run k times.
The sequence is 3,5,9,17,33……2^n+1.
2+1,4+1,8+1,16+1,32+1…….
2^n+1=2*2^k+1
n=k+1
k=n-1;
First loop will run n times;
So, the time complexity is O(n*(n-1))->O(n^2).

Answer to the question no-04

In this code , the second 'for' loop has some problematic issues. The loop starts from 0. But in the 'if' statement there is a problem. In an array, the index can not be less than 0. If it occurs in some cases, a runtime error is going to happen. In that statement a[i-1] is invalid when i=0. So, there is a flaw in this code. The correct code is:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
        int n;
        cin>>n;
```

```
        vector<int>a(n);
        for(int i=0;i<n;i++)
                cin>>a[i];
        sort(a.begin(),a.end());
       int ans = 0;
        for(int i=0 ; i<=n ; i++){
                if(i==0){ans++;continue;}
                if(a[i]!=a[i-1]){
                        ans++;}}
        cout<<ans;
        return 0;
}
```

Answer to the question no-05

```
vector<int>d[n+1];
for(int i=1 ; i<=n ; i++)
        for(int j=i ; j<=n ; j = j+i )
                d[j].push_back(i);
```

There is a nested loop in that code.
First loop will run n times. Then
If we assume
i=1; then the 2nd loop will run-> n times.
i=2, the 2nd loop ->n/2.
i=3, the 2nd loop -> n/3.
……..
i=n, the 2nd loop -> n/n.
So, time complexity is O(n*n/n)->O(n).
Space complexity is O(1+n+1)->O(n).

Answer to the question no-06

Name	Accessibility from own class	Accessibility from derived class	Accessibility from world
Public	YES	YES	YES
Private	YES	NO	NO

Protected	YES	YES	NO

Answer to the question no-07

new and delete are 2 keywords of C++. Both are used to create memory allocation and remove that allocation.

new-> when we use this keyword we can get some free memory from ram. As a result we get some space if there is free memory available. This allocates the memory dynamically. And the work is done perfectly.

delete-> this keyword is used to free the allocated memory. As we have created int* p=new int. So, after using *p we have to use delete p to free the memory. It helps to run the program for a long time without any flaws.

Answer to the question no-08

Yes, I agree with Bob.

Time complexity is O(n^3);

n=10^6

So, total works=10^6*3=10^18.

Time to complete =10^18/10^9=10^9s

We know 1 year = 365*24*60*60=31536000s

So, the time is to complete that=10^9/31536000=31.71 years (approximately).

Answer to the question no-09

Linear Search	Binary Search
1. Time complexity O(n).	1. Time Complexity O(log(n)).
2. It starts from the beginning of an array.	2. It starts from the mid point of the array and run until the searching value is found.

Answer to the question no-10

Yes, there is a flaw in this function. We have created a new int from ram. As a result, it takes some memory from ram. After using this memory we have to remove that value or delete for freeing the space.
The right code is:

```
void func()
{
    int* p = new int;

    delete p;
    return;
}
```