Question No. 01

Write down all the steps of Bubble Sort on the Following Array.

Index	0	1	2	3	4	5
Value	7	2	13	2	11	4

For example:

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
……..

Answer 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

Total 5 swaps done in 1st iteration and now the largest value is in sorted position

2nd iteration:

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

Total 3 swaps done in 2nd iteration and now the second largest value is in sorted position

3rd iteration:

1st step: 2 2 **7 4** 11 13 -> 2 2 4 7 11 13

Total 1 swap done in 3rd iteration and now the third largest value is in sorted position

Now at this point the array is sorted. The final sorted array is [2 2 4 7 11 13].

Question No. 02
Write down two differences between array and vector in C++.

Answer No. 02

Array	**Vector**
Array stores a fixed size sequential collection of elements of same data type which is index based	Vector is a sequential container to store elements and it’s not index based
Array is memory efficient	Vector needs more memory

Question No. 03

Write down the time complexity with proper explanation of the following code segment.

```
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.

Answer No. 03

The time complexity of this code is $O(n^2)$.
Here, the outer loop runs n times and the inner loop runs n times, so total n*n iterations.

The builtin_popcount function counts the number of set bits in an integer. Now the outer loop will iterate n times always but when builtin_popcount function returns 2 then only then the inner loop iterates n times. So at the worst case it will iterates n*n times.

So, we can say that the time complexity of this code is $O(n^2)$.

Question No. 04

Look at the following code which calculates the number of distinct elements. Are there any flaws in this code? If yes, write down the flaws with proper explanation.

```
#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(a[i]!=a[i-1])
                        ans++;
        cout<<ans;
        return 0;
}
```

Answer No. 04

Yes! There is a flow in this code.
In this code the loop that counts the number of distinct elements will not work correctly because it compares each element to the element before it in the sorted array. Now here, the loop starts at i = 0. So, the first element in the array will be compared to an element that does not exist because it is a[0-1] = a[-1]. That will cause an error.

So, we have to start the loop from 1. Here i = 1 has to be initialized.

Here is the modified code:

```
#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=1 ; i<=n ; i++) // Starting the loop at i = 1
          if(a[i]!=a[i-1])
                ans++;
    cout<<ans;
    return 0;
}
```

Question No. 05

Write down the time and space complexity with proper explanation of the following code segment.

```
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);
```

Answer No. 05

The time complexity of this code is $O(n^2)$.

Here, the outer loop runs n times. In the first iteration the inner loop runs n times also but after that it decreases by 1. So, we can say that first time the total code executes n*n instructions. After it the other iterations will not be that significant. It executes total n*(n-1)/2 instructions where we can take n*n and the other values we can ignore. So, the time complexity will be $O(n^2)$.

The space complexity of this code is O(n).

The array d[] has n elements, where each of them is a vector of integers, so the space required is (n*size of vector<int>). Since the size of a vector is proportional to the number of elements it contains, and each element is an integer, the space required for each element is sizeof(int). So, the space complexity will be O(n).

Question No. 06

Fill up the following table with ‘YES’ or ‘NO’ in each cell in the context of public, private and protected access modifiers in C++ Class. First cell is already filled up for your convenience.

Name	Accessibility from own class	Accessibility from derived class	Accessibility from world
Public	YES		
Private			
Protected			

Answer 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**

Question No. 07
What is ‘new’ and ‘delete’ in C++.
Answer No. 07

new is an operator that denotes a request for memory allocation on the free space. If sufficient memory is available, a new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.
Example:
void func()
{
int* p = new int;
return;
}
 	
delete is an operator which is used to free the dynamically allocated memory. This operator deallocates dynamically allocated memory.
Example:
void func()
{
int* p = new int;
delete p;
return;
}

Question No. 08
Alice wrote a new algorithm which works in $O(n^3)$ where n can be atmost 10^6. Bob told Alice that it will take years to finish in the worst case. Do you agree with Bob? If yes, then approximately how many years will it take to finish? Assume Alice's computer can run 10^9 instructions in 1 second.
Answer No. 08
Yes! I agree with Bob. It will take years to finish in the worst case. Here the time complexity is $O(n^3)$. In the worst case $n = 10^6$. So, the number of instructions the algorithm will need to execute is $(10^6)^3 = 10^{18}$ Now, Alice's computer can run 109 instructions per second. So, Here, 10^9 instructions to execute in 1 second 1 instructions to execute in $1/10^9$ second 10^{18} instructions to execute in $10^{18}/10^9$ seconds = 10^9 second Now, There are 31536000 seconds in a year, it will take a computer approximately $10^9/31536000$ = **31.71 years** to execute the algorithm in the worst case.

Question No. 09

Write down two differences between binary search and linear search.

Answer No. 09

Binary search	**Linear search**
In binary search input data need to be in sorted order	In linear search input data need not to be in sorted order.
Time complexity O(log n)	Time complexity O(n)

Question No. 10

Suppose you wrote a code for a server which contains the following function.

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

Are there any flaws in the function? If yes, then explain the flaw and modify the function so that there is no flaw in the function. You cannot delete any lines of code from the function. You are only allowed to write your own code to overcome the flaw.

Answer No. 10

Yes! There is a flaw in the function.

The flaw is that the new operator dynamically allocates memory on the heap, but the allocated memory wasn't freed before returns.

Modified function:

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