Answer Script

Question No. 01

1. 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
End of step: 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
End of step: 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
End of step: 2 2 4 7 11 13

4th iteration:
1st step: 2 2 4 7 11 13 -> 2 2 4 7 11 13
2nd step: 2 2 4 7 11 13 -> 2 2 4 7 11 13
End of step: 2 2 4 7 11 13

5th iteration:
1st step: 2 2 4 7 11 13 -> 2 2 4 7 11 13
End of step: 2 2 4 7 11 13

6th iteration:
Final step: 2 2 4 7 11 13

Question No. 02

2. Write down two differences between array and vector in C++.

Answer No. 02

Array	Vector
Arrays can be implemented in a static or dynamic way.	Vectors can only be implemented dynamically.
Arrays have a fixed size.	Vectors have a dynamic size. They can resize themselves.
In array, no methods are provided for adding and removing elements.	In vector, methods are provided for adding and removing elements.
Array is unsynchronized.	Vector is synchronized.
Array is not a class.	Vector is a class.
Fixed-size memory allocates by the array.	In vector, memory allocates by the dynamic.
Index-based data structures.	Vectors are not index-based data structures.
Available in both C and C++.	Only available in C++.
Reallocation of memory in case of Array is not done implicitly.	Reallocation of memory in the case of Vectors is done implicitly.
In programming, arrays can never be copied or assigned directly.	In programming, vectors can be copied or assigned directly.

Question No. 03

3. 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

Here, this code time complexity is O(n^2).

1st we see this is nested for loop. One is the inner loop and another one is the outer loop. The outer loop works n times. So, it's time complexity O(n).

Inside the outer loop there is a condition. In this condition has a function name builtin_popcount(i). This function returns a value/number. If the value is equal to 2 then it goes inside. Inside has a for loop. This loop works n times. So, it's time complexity O(n).

So, total time complexity is O(n*n) or O(n^2).

Question No. 04

4. 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 are flaws in this code.

Calculates the number of distinct elements using a for loop. In this for loop first i initialise 0. It will not be 0. It will be 1. This for loop work 1 to n. Every times i value increases by one.

Correct 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++)
        if(a[i]!=a[i-1])
            ans++;

    cout<<ans;
    return 0;
}
```

Question No. 05

5. 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

In this code,
The time complexity: O(n log n)
The space complexity: O(n)

Space complexity O(n). Because here declare n size vector array. It stores n elements that's why it's space complexity O(n).

This is a nested loop. Here are two loops: outer loop and inner loop.

For: i = 1

j = i; j<=n; j=j+i;

1. j = 1; j<=n; j=1+1;
2. j = 2; j<=n; j=2+1;
3. j = 3; j<=n; j=3+1;

If i = 1 the time complexity: O(n)

For: i = 2

j = i; j<=n; j=j+i;

1. j = 2; j<=n; j=2+2;
2. j = 4; j<=n; j=4+2;
3. j = 6; j<=n; j=6+2;

If i = 2 the time complexity: O(n/2)

For: i = 3

j = i; j<=n; j=j+i;

1. j = 3; j<=n; j=3+3;
2. j = 6; j<=n; j=6+3;
3. j = 9; j<=n; j=9+3;

If i = 3 the time complexity: O(n/3)

For: i = 4

j = i; j<=n; j=j+i;

1. j = 4; j<=n; j=4+4;
2. j = 8; j<=n; j=8+4;
3. j = 12; j<=n; j=12+4;

If i = 4 the time complexity: O(n/4)

So, n + n/2 + n/3 + n/4 + n/5 ……..+ n/n

= n [1 + 1/2 + 1/3 + 1/4 + 1/5 + ……..+ 1/n]

= n log n

The time complexity: O(n log n)

Question No. 06

6. 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	YES	YES
Private	YES	NO	NO
Protected	YES	YES	NO

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

new:

The new operator is used to allocate memory for a variable or any other entity like objects or arrays on a heap memory area. If a sufficient amount of memory is available on the heap, the new operator will initialise the memory and return the address of the newly allocated memory and you can use pointers to store the address of that memory location.

Syntax for new operator:

pointer-variable = new data-type;

Here, the pointer variable is the pointer of type data-type. Data type could be any built-in data type including array or any user-defined data type including structure and class.

delete:

Since the programmer has allocated memory at runtime, it’s the responsibility of the programmer to delete that memory when not required. So at any point, when programmers feel a variable that has been dynamically allocated is no longer required, they can free up the memory that it occupies in the free store or heap with the “delete” operator. It returns the memory to the operating system. This is also known as memory deallocation. Also, memory will be deallocated automatically once the program ends.

Syntax for delete operator:

delete pointer_variable;

Question No. 08

8. 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

Here, The time complexity: O(n^3)
Given,
$n = 10^6$
$n^3 = 10^{18}$

1000000000 instructions run in 1 second
1 instructions run in 1/1000000000 second
1000000000000000000 instructions run in 1000000000000000000/1000000000 second
= 1000000000 seconds

1000000000 seconds
= (1000000000÷60) minutes
= 16,666,666.666 minutes
= (16,666,666.666 ÷ 60) hours
= 277,777.777 hours
= (277,777.777 ÷ 24) days
= 11,574.074 days
= (11,574.074 ÷ 30) months
= 385.802 months
= (385.802 ÷ 12) years
= 32.15 years.

Question No. 09

9. Write down two differences between binary search and linear search.

Answer No. 09

Linear Search	Binary Search
In a linear search, the elements don't need to be arranged in sorted order.	The pre-condition for the binary search is that the elements must be arranged in a sorted order.
It is based on the sequential approach.	It is based on the divide and conquer approach.
Less Complex to implement.	More Complex to implement.
It is less efficient in the case of large-size data sets.	It is more efficient in the case of large-size data sets.
It can be implemented on both a single and multidimensional array.	It can be implemented only on a multidimensional array.
It is preferable to use it with small-sized data sets.	It is used with large sized datasets.
Linear search is a slow process.	Binary search is comparatively faster.
It can be implemented on any linear data structure like an array, linked list.	It can only be used on data structures that have two-way traversal.
The worst-case time complexity is O(n).	The worst case time complexity is O(log

Question No. 10

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 are flaws in the function.

Given function:

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

In this function the return type is void. We know void means it doesn't return any value. But here we can see the return keyword. We should return something and change the return type void to int.

Here, memory is allocated dynamically. We know if we declare any memory by new we should delete it when our work is finished. Here, memory doesn't delete when this function work is finished.

My Code:

```
void func()
{
    int* p = new int;
   *p = 10 // we can store anything here
    delete p;
}
```

or

```
int func()
{
    int* p = new int;
   *p = 10 // we can store anything here
    return  *p; // it sent p value which is 10
}
```