Question 1

Data Structure, which stores a fixed-size sequential collection of elements of the same type.?

A	loop
B	array
C	exception
D	int

✔ **Your Answer is Correct**
Correct Answers: B
Your Answers: B

Question 2

An array is used to store a collection of data.

A	true
B	false
C	either be
D	cant say

✔ **Your Answer is Correct**
Correct Answers: A
Your Answers: A

Question 3

All arrays consists of _____ memory locations.

A	simple
B	contiguous
C	distant
D	cant say

✔ **Your Answer is Correct**
Correct Answers: B
Your Answers: B

Question 4

The lowest address of array is?

A	0
B	1
C	2
D	3

✔ **Your Answer is Correct**
Correct Answers: A
Your Answers: A

Question 5

The arraySize must be an integer constant greater than ?

A	0
B	1
C	2
D	3

✔ **Your Answer is Correct**
Correct Answers: A
Your Answers: A

Question 6

The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ].

A	true
B	false
C	cant say
D	don't know

✘ **Your Answer is Incorrect**
Correct Answers: A
Your Answers: C

Question 7

Which of the following correctly declares an array in C++?

A	array[10];
B	array array[10];
C	int array;
D	int array[10];

✔ **Your Answer is Correct**
Correct Answers: D
Your Answers: D

Question 8

What is the index number of the last element of an array with 9 elements?

A	0
B	1
C	8
D	9

✔ **Your Answer is Correct**
Correct Answers: C
Your Answers: C

Which of the following gives the memory address of the first element in array?

A	arr
B	arr+1
C	arr+2
D	arr-1

✔ **Your Answer is Correct**
Correct Answers: A
Your Answers: A

What is the Output of following program?

```
#include <stdio.h>
using namespace std;
int array1[] = {1200, 200, 2300, 1230, 1543};
int array2[] = {12, 14, 16, 18, 20};
int temp, result = 0;
int main()
{
for (temp = 0; temp < 5; temp++) {
result += array1[temp];
}
for (temp = 0; temp < 4; temp++) {
result += array2[temp];
}
cout << result;
```

```
{
for (temp = 0; temp < 5; temp++) {
result += array1[temp];
}
for (temp = 0; temp < 4; temp++) {
result += array2[temp];
}
cout << result;
return 0;
}
```

A	6553
B	6533
C	6522
D	12200

✔ **Your Answer is Correct**

Question 11

What is Time and Space Complexity of following code?

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
a = a + rand();
}
for (j = 0; j < M; j++) {
b = b + rand();
}
```

A	O(N * M) time, O(1) space
B	O(N + M) time, O(N + M) space
C	O(N + M) time, O(1) space
D	O(N * M) time, O(N + M) space

✗ Your Answer is Incorrect
Correct Answers: C
Your Answers: A

Question 12

What is Time and Space Complexity of following code?

```
int a = 0;
for (i = 0; i < N; i++) {
for (j = N; j > i; j--) {
a = a + i + j;
}
}
```

A	O(N)
B	O(N*log(N))
C	O(N * Sqrt(N))
D	O(N*N)

```
for (j = N; j > i; j--) {
a = a + i + j;
}
}
```

A	O(N)
B	O(N*log(N))
C	O(N * Sqrt(N))
D	O(N*N)

✗ Your Answer is Incorrect
Correct Answers: D
Your Answers: C

What is Time and Space Complexity of following code?

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
for (j = 2; j <= n; j = j * 2) {
k = k + n / 2;
}
}
```

A	O(N)
B	O(N*log(N))
C	O(n^2)
D	O(n^2Logn)

.

A	O(N)
B	O(N*log(N))
C	O(n^2)
D	O(n^2Logn)

✖ Your Answer is Incorrect

Correct Answers: B

Your Answers: C

Explanation: If you notice, j keeps doubling till it is less than or equal to n. Several times, we can double a number till it is less than n would be log(n).
Let's take the examples here.
for n = 16, j = 2, 4, 8, 16
for n = 32, j = 2, 4, 8, 16, 32
So, j would run for O(log n) steps.
i runs for n/2 steps.
So, total steps = O(n/ 2 * log (n)) = O(n*logn)

Question 14

What is Time and Space Complexity of following code?

```
int a = 0, i = N;
while (i > 0) {
a += i;
i /= 2;
}
```

A	O(N)
B	O(N*log(N))
C	O(n^2)
D	O(LogN)

✔ **Your Answer is Correct**
Correct Answers: D

Question 15

Algorithm A and B have a worst-case running time of O(n) and O(logn), respectively. Therefore, algorithm B always runs faster than algorithm A.

A	true
B	false
C	don't know
D	any one

✘ **Your Answer is Incorrect**
Correct Answers: B
Your Answers: A

Explanation: The Big-O notation provides an asymptotic comparison in the running time of algorithms. For n < n0, algorithm A might run faster than algorithm B, for instance.