1. In C++, the index of the first element in an array is:

a) -1

b) 0

**c) 1**

d) None of the above

2. #include <iostream>

using namespace std;

int main() {

int arr[] = {1, 2, 3, 4, 5};

cout << arr[3] << endl;

return 0;

}

What will be the output of the above code?

a) 1

b) 2

c) 3

**d) 4**

3. The time complexity to access an element in an array by its index is:

a) O(n)

**b) O(1)**

c) O(log n)

d) O(n^2)

4. #include <iostream>

using namespace std;

int main() {

int arr[] = {3, 6, 9, 12, 15};

cout << \*(arr + 2) << endl;

return 0;

}

What will be the output of the above code?

a) 3

b) 6

**c) 9**

d) 12

5. When an element is deleted from an array, the remaining elements are shifted to fill the empty space. The time complexity for deletion from the end of the array is:

a) O(1)

**b) O(1) (amortized)**

c) O(n)

d) O(log n)

6.

#include <iostream>

using namespace std;

int main() {

int arr[5] = {1, 2, 3, 4, 5};

cout << arr[6] << endl;

return 0;

}

What will be the output of the above code?

a) 5

b) 4

c) 3

**d) Garbage value**

7. The time complexity of the Bubble Sort algorithm for sorting an array of n elements is:

a) O(n)

b) O(log n)

**c) O(n^2)**

d) O(n log n)

8. #include <iostream>

using namespace std;

int main() {

int arr[] = {3, 8, 1, 6, 4};

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

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

What will be the output of the above code?

**a) 3 8 1 6 4**

b) 4 6 1 8 3

c) 3 6 1 8 4

d) 1 3 4 6 8

9. Binary Search is more efficient than Linear Search because:

a) Binary Search uses less memory.

b) Binary Search has a lower time complexity.

**c) Binary Search reduces the search space by half with each iteration.**

d) Binary Search is a recursive algorithm.

10. #include <iostream>

using namespace std;

int main() {

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

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

int count = 0;

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (arr[i] == arr[j]) {

count++;

break;

}

}

}

cout << count;

return 0;

}

What will be the output when the above code is executed?

**a) 3**

b) 4

c) 5

d) 6

11. Which data structure can be used to efficiently implement a stack using an array?

**a) One-dimensional array**

b) Two-dimensional array

c) Linked List

d) Queue

12.

#include <iostream>

using namespace std;

int main() {

int arr1[] = {1, 2, 3, 4, 5};

int arr2[] = {10, 20, 30, 40, 50};

int n = sizeof(arr1) / sizeof(arr1[0]);

for (int i = 0; i < n; i++) {

cout << arr1[i] + arr2[i] << " ";

}

return 0;

}

What will be the output of the above code?

a) 1 2 3 4 5

b) 10 20 30 40 50

**c) 11 22 33 44 55**

d) 15 15 15 15 15

13. Which operation is used to access all the elements of an array in sequence?

a) Insertion

b) Deletion

**c) Traversal**

d) Sorting

14. #include <iostream>

using namespace std;

void merge(int arr1[], int m, int arr2[], int n, int merged[]) {

int i = 0, j = 0, k = 0;

while (i < m && j < n) {

if (arr1[i] < arr2[j]) {

merged[k++] = arr1[i++];

} else {

merged[k++] = arr2[j++];

}

}

while (i < m) {

merged[k++] = arr1[i++];

}

while (j < n) {

merged[k++] = arr2[j++];

}

}

int main() {

int arr1[] = {1, 3, 5, 7, 9};

int arr2[] = {2, 4, 6, 8, 10};

int m = sizeof(arr1) / sizeof(arr1[0]);

int n = sizeof(arr2) / sizeof(arr2[0]);

int merged[m + n];

merge(arr1, m, arr2, n, merged);

for (int i = 0; i < m + n; i++) {

cout << merged[i] << " ";

}

return 0;

}

What will be the output of the above code?

a) 1 3 5 7 9

b) 2 4 6 8 10

**c) 1 2 3 4 5 6 7 8 9 10**

d) Compiler error

15. Which sorting algorithm has the best time complexity in the average case?

a) Bubble Sort

**b) Merge Sort**

c) Insertion Sort

d) Quick Sort

16. #include <iostream>

using namespace std;

void insertionSort(int arr[], int n) {

for (int i = 1; i < n; i++) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

int main() {

int arr[] = {5, 2, 9, 1, 5};

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

insertionSort(arr, n);

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

What will be the output of the above code?

a) 5 2 9 1 5

**b) 1 2 5 5 9**

c) 9 5 5 2 1

d) 1 2 5 5 9

17. The process of arranging elements of an array in a specific order is called:

a) Traversal

b) Insertion

**c) Sorting**

d) Deletion

18. #include <iostream>

using namespace std;

int main() {

int arr[] = {5, 2, 8, 3, 1};

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

int key = 3;

int index = -1;

for (int i = 0; i < n; i++) {

if (arr[i] == key) {

index = i;

break;

}

}

cout << index;

return 0;

}

What will be the output when the above code is executed with key = 3?

**a) 3**

b) 4

c) 2

d) -1

19. The time complexity for merging two sorted arrays of size m and n into a third array of size m+n is:

**a) O(m + n)**

b) O(m \* n)

c) O(m log n)

d) O(log(m + n))

20. #include <iostream>

using namespace std;

int main() {

int arr[] = {5, 3, 8, 1, 2, 7};

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

for (int i = 0; i < n; i++) {

int temp = arr[i];

arr[i] = arr[n - 1 - i];

arr[n - 1 - i] = temp;

}

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

What will be the output when the above code is executed?

a) 7 2 1 8 3 5

**b) 5 3 8 1 2 7**

c) 7 5 2 1 8 3

d) 2 5 1 8 3 7