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[5];

cout << sizeof(arr) / sizeof(arr[0]) << endl;

return 0;

}

What will be the output of the above code?

a) 1

b) 2

c) 3

**d) 5**

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

cout << arr << endl;

return 0;

}

What will be the output of the above code?

a) 2

b) 4

c) 6

**d) Memory address of arr[0]**

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 beginning of the array is:

a) O(1)

**b) O(n) (amortized)**

c) O(n)

d) O(log n)

6. #include <iostream>

using namespace std;

int main() {

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

int\* ptr = arr;

cout << \*(ptr + 3) << endl;

return 0;

}

What will be the output of the above code?

a) 1

b) 2

c) 3

**d) 4**

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

a) O(n)

b) O(log n)

**c) O(n log n)**

d) O(n^2)

8. #include <iostream>

using namespace std;

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

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

int main() {

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

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

bubbleSort(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 9 5

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

a) Binary Search uses less memory.

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

c) Linear Search is only applicable to smaller arrays.

d) Binary Search works on unsorted arrays.

10. #include <iostream>

using namespace std;

int main() {

int arr[] = {4, 2, 7, 1, 3};

int key = 7;

int low = 0, high = 4, mid;

while (low <= high) {

mid = (low + high) / 2;

if (arr[mid] == key) {

cout << "Found" << endl;

break;

}

else if (arr[mid] < key) {

low = mid + 1;

}

else {

high = mid - 1;

}

}

return 0;

}

What will be the output of the above code if the key is 7?

**a) Found**

b) Not Found

c) Compiler error

d) Undefined behavior

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

a) One-dimensional array

**b) Circular array**

c) Two-dimensional array

d) Linked List

12. #include <iostream>

using namespace std;

int main() {

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

int arr2[5];

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

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

arr2[i] = arr1[i] \* 2;

}

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

cout << 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) 2 4 6 8 10**

d) 3 6 9 12 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 selectionSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

int minIndex = i;

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

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

int temp = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = temp;

}

}

int main() {

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

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

selectionSort(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**

15. Which sorting algorithm has the best time complexity in the average case and requires additional space for merging?

a) Bubble Sort

b) Quick Sort

**c) Merge Sort**

d) Insertion Sort

16. #include <iostream>

using namespace std;

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

int sum = 0;

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

sum += arr[i];

}

cout << sum;

}

int main() {

int arr[] = {4, 7, 2, 9, 1};

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

findSum(arr, n);

return 0;

}

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

a) 15

b) 21

**c) 23**

d) 32

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[] = {10, 3, 7, 1, 4};

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

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

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

arr[j + 1] = 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) 1 3 4 7 10**

b) 10 7 4 3 1

c) 1 4 3 7 10

d) 10 3 7 1 4

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;

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

for (int i = 0; i < n - 1; i++) {

int minIndex = i;

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

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

int temp = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = temp;

}

}

int main() {

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

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

selectionSort(arr, n);

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

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

}

return 0;

}

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

**a) 1 2 3 6 9**

b) 9 6 3 2 1

c) 9 2 6 3 1

d) 1 3 2 6 9