1. Heap Sort is a sorting algorithm based on the concept of a \_\_\_\_\_\_\_.

a) Linked list

**b) Priority queue**

c) Hash table

d) Binary search tree

2. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

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

make\_heap(arr, arr + n);

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

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

}

return 0;

}

a) 13 12 7 5 6 11

**b) 13 11 12 5 6 7**

c) 13 11 12 6 5 7

d) 7 6 5 13 11 12

3. Which of the following statements is true about Heap Sort?

**a) It has a worst-case time complexity of O(n log n).**

b) It is an in-place sorting algorithm.

c) It is a stable sorting algorithm.

d) It is suitable for sorting large linked lists.

4. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

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

make\_heap(arr, arr + n);

sort\_heap(arr, arr + n);

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

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

}

return 0;

}

**a) 5 6 7 11 12 13**

b) 13 12 11 7 6 5

c) 5 6 7 12 11 13

d) 13 11 12 7 6 5

5. Heap Sort works by first building a \_\_\_\_\_\_\_ from the given array.

a) Binary search tree

b) Min-heap

**c) Max-heap**

d) AVL tree

6. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

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

make\_heap(arr, arr + n);

pop\_heap(arr, arr + n);

n--;

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

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

}

return 0;

}

**a) 12 11 7 5 6**

b) 13 11 12 7 6

c) 12 11 6 5 7

d) 11 5 6 7 12

7. During the heapification process in Heap Sort, the maximum element is moved to the \_\_\_\_\_\_\_.

**a) Root of the heap**

b) Rightmost leaf node

c) Leftmost leaf node

d) Middle element of the heap

8. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

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

make\_heap(arr, arr + n);

pop\_heap(arr, arr + n);

n--;

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

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

}

cout << endl;

sort\_heap(arr, arr + n);

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

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

}

return 0;

}

A) 11 6 7 5

5 6 7 11

B) 13 11 12 7

7 11 12 13

**C) 12 11 7 5 6**

**5 6 7 11 12**

D) 11 5 6 7

5 6 7 11

9. What is the time complexity of Heap Sort for sorting n elements in the worst case?

a) O(n)

**b) O(n log n)**

c) O(log n)

d) O(n^2)

10. Choose the correct option to fill? X so that the code given below implements the Heap sort.

#include <stdio.h>

void heapify(int arr[], int n, int i)

{

int largest = i; // Initialize largest as root

int l = 2\*i + 1; // left = 2\*i + 1

int r = 2\*i + 2; // right = 2\*i + 2

if (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

if (largest != i)

{

swap(arr[i], arr[largest]);

heapify(arr, n, largest);

}

}

void heapSort(int arr[], int n)

{

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i=n-1; i>=0; i--)

{

X;

heapify(arr, i, 0);

}

}

void printArray(int arr[], int n)

{

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

printf(“%d”,arr[i]);

printf(“\n”);

}

int main()

{

int arr[] = {12, 11, 13, 5, 6, 7};

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

heapSort(arr, n);

printf(“Sorted array is \n");

printArray(arr, n);

}

a) swap(arr[0], arr[n])

b) swap(arr[i], arr[n])

**c) swap(arr[0], arr[i])**

d) swap(arr[i], arr[2\*i])

11. In a Max-heap, the element with the highest value is always located at the \_\_\_\_\_\_\_.

**a) Root of the heap**

b) Rightmost leaf node

c) Leftmost leaf node

d) Middle element of the heap

12. What is the output of the following code snippet?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(10);

pq.push(20);

pq.push(5);

pq.pop();

cout << pq.top();

return 0;

}

a) 5

**b) 10**

c) 20

d) Compilation Error

13. When performing a deletion in a Max-heap, after removing the root element, which element replaces it to maintain the heap property?

a) Largest element from the left subtree

b) Smallest element from the right subtree

**c) Largest element from the right subtree**

d) Smallest element from the left subtree

14. What will be the output of the following C++ code?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(50);

pq.push(20);

pq.push(30);

while (!pq.empty()) {

cout << pq.top() << " ";

pq.pop();

}

return 0;

}

a) 20 30 50

**b) 50 30 20**

c) 30 20 50

d) Compilation Error

15. After deleting an element from a Max-heap, the remaining elements need to be rearranged to maintain the \_\_\_\_\_\_\_.

a) Binary search tree property

b) Min-heap property

**c) Max-heap property**

d) Balanced tree property

16. What is the output of the following C++ code?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(10);

pq.push(15);

pq.push(5);

pq.push(25);

pq.pop();

cout << pq.top();

return 0;

}

a) 5

b) 10

**c) 15**

d) 25

17. What is the time complexity of deleting an element from a Max-heap with n elements?

**a) O(log n)**

b) O(n)

c) O(n log n)

d) O(1)

18. What will be the output of the following C++ code?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(30);

pq.push(10);

pq.push(20);

cout << pq.top() << " ";

pq.pop();

cout << pq.top();

return 0;

}

a) 10 20

b) 20 10

**c) 30 20**

d) Compilation Error

19. Heap Deletion is commonly used to efficiently implement the \_\_\_\_\_\_\_ operation in priority queues.

a) Insertion

b) Deletion

**c) Peek**

d) Search

20. What is the output of the following code snippet?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(5);

pq.push(10);

pq.push(15);

pq.push(10);

while (!pq.empty()) {

cout << pq.top() << " ";

pq.pop();

}

return 0;

}

a) 5 10 10 15

**b) 15 10 10 5**

c) 5 10 15 10

d) Compilation Error