C++ Cheat Sheet for LeetCode & Competitive Programming

1. Arrays

   - Declaration & Initialization:

```
int arr[5];

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

   - 2D Arrays:

```
int matrix[3][3];

int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

2. Vectors (Dynamic Array)

   - Declaration:

```
vector<int> vec;

vector<int> vec(n, 0); // Size n, initialized with 0
```

   - Common Operations:

```
vec.push_back(5); vec.size(); sort(vec.begin(), vec.end());
```

3. Nodes for Trees, Graphs, and Linked Lists

   - Binary Tree Node:

```
struct TreeNode {

    int val;

    TreeNode* left;

    TreeNode* right;

    TreeNode(int x) : val(x), left(NULL), right(NULL) {}
```

```
  };
```

- Graph Node:

```
  struct GraphNode {
      int val;
      vector<GraphNode*> neighbors;
      GraphNode(int x) : val(x) {}
  };
```

- Linked List Node:

```
  struct ListNode {
      int val;
      ListNode* next;
      ListNode(int x) : val(x), next(NULL) {}
  };
```

4. Custom Comparators

- Custom Comparator for Sorting:

```
  sort(arr, arr + n, [](int a, int b) { return a > b; });
```

- Custom Comparator for Priority Queue (Min-Heap):

```
  priority_queue<int, vector<int>, greater<int>> minHeap;
```

- Priority Queue with Custom Comparator for Pair:

```
  priority_queue<pair<int, int>, vector<pair<int, int>>, CustomCompare> pq;
  struct CustomCompare {
        bool operator()(pair<int, int> a, pair<int, int> b) {
            return a.first > b.first;  // Custom condition
        }
    };
```

5. Stack, Queue, and Deque

- Stack:

```
stack<int> st;
st.push(5); st.pop();
```

- Queue:

```
queue<int> q;
q.push(5); q.front(); q.pop();
```

- Deque:

```
deque<int> dq;
dq.push_front(1); dq.push_back(2);
```

6. Maps and Sets

- Map: map<int, string> mp;

```
mp[1] = "One";
```

- Set: set<int> s;

```
s.insert(5); s.count(5);
```

7. Math Functions

- Absolute: abs(x);

- Max/Min: max(a, b); min(a, b);

- Power: pow(base, exponent);