Initialisation :- **vector** < *data type* > *var\_name* ( vector <int> v1 )  
 2D-Array => **(** vector <int> v1[] **)**

v1.begin() - Returns an iterator pointing to the first element in the vector  
v1.end() - Returns an iterator pointing to the theoretical element that follows last element in the vector

v1.rbegin() –Returns a reverse iterator pointing to the last element in the vector (reverse beginning).   
 It moves from last to first element  
v1.rend() – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector   
 (considered as reverse end)

v1[pos] - Element at position (pos)   
v1.at(pos) - Element at position (pos)   
v1.front( ) - Element at front (pos)  
v1.back( ) - Element at end (pos)

v1.assign(5, 10) - Assigns value of 10 to 5 elements  
v1.assign(g1, g1 + 2) - Assigns value from pos 0 to pos 2 of array g1

v1.push\_back(10) - Pushes value of 10 at the end of the vector and increases size by 1 .  
v1.pop\_back( ) - Pops the element at the end of the vector, decreases the vector container size by 1

v1.insert( it , 20 ) - Inserts element 20 at position before ‘it’ and then returns an iterator pointing to 20  
v1.insert( it , 10 , 20 ) - Inserts “10” 20s at positions before ‘it’ and then returns an iterator pointing to the 1st 20  
v1.insert( it , itL , itR ) - Inserts elements starting at itL to itR (inclusive?) after ‘it’ *inclusive* ( NO RETURN )

v1.erase(it) - Erases element referenced by the position ‘it’ and returns iterator followed by deleted element   
v1.erase(itL , itR) - Erases elements in range [itL , itR) and return iterator to next element , I guess itR hoga who

v1.swap(v2 ) - Swaps elements in vector v1 with v2   
v1.clear() - Clears vector v1

v1.lower\_bound ( itL , itR , val ) - returns an iterator pointing to the first element in the range [ itL , itR) which is   
 equivalent or after ‘val’.  
v1.upper\_bound ( itL , itR , val ) - returns an iterator pointing to the first element in the range [ itL , itR) which is   
 after ‘val’.