**C++ Programming Questions and Answers – Sequence Adapters**

1. What do container adapter provide to interface?  
a) Restricted interface  
b) More interface  
c) No interface  
d) Memory interface

Answer: a  
Explanation: A container adapter provides a restricted interface to a container.In particular, adapters do not provide iterators; they are intended to be used only through their specialized interfaces.

2. What does the sequence adaptor provide?  
a) Insertion  
b) Deletion  
c) Interface to sequence container  
d) Insertion & Deletion

Answer: c  
Explanation: Sequence adaptor provides interface to sequence container.

3. Which are presented in the container adaptors?  
a) stack  
b) queue  
c) priority\_queue  
d) all of the mentioned

Answer: d  
Explanation: These mentioned things are presented in container adapters.

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

1. #include <iostream>
2. #include <queue>
3. using namespace std;
4. int main ()
5. {
6. queue<int> myqueue;
7. myqueue.push(12);
8. myqueue.push(75);
9. myqueue.back() -= myqueue.front();
10. cout << myqueue.back() << endl;
11. return 0;
12. }

a) 12  
b) 75  
c) 63  
d) 74

Answer: c  
Explanation: In this program, We used the queue operation and performed the back operation. Because of that operation, We got the output as 63.  
Output:

$ g++ sca.cpp

$ a.out

63

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

1. #include <iostream>
2. #include <queue>
3. using namespace std;
4. int main ()
5. {
6. queue<int> myqueue;
7. int sum (0);
8. for (int i = 1; i <= 10; i++)
9. myqueue.push(i);
10. while (!myqueue.empty())
11. {
12. sum += myqueue.front();
13. myqueue.pop();
14. }
15. cout << sum << endl;
16. return 0;
17. }

a) 51  
b) 52  
c) 54  
d) 55

Answer: d  
Explanation: In this program, We used the push and pop operation of quueue to find out the total of all the number from 1 to 10.  
Output:

$ g++ sca1.cpp

$ a.out

55

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

1. #include <iostream>
2. #include <queue>
3. using namespace std;
4. int main ()
5. {
6. priority\_queue<int> mypq;
7. mypq.push(30);
8. mypq.push(100);
9. mypq.push(25);
10. mypq.push(40);
11. while (!mypq.empty())
12. {
13. cout << " " << mypq.top();
14. mypq.pop();
15. }
16. cout << endl;
17. return 0;
18. }

a) 100 40 30 25  
b) 100 40 30  
c) 100 40  
d) 100 30 25

Answer: a  
Explanation: In this program, We used priority\_queue and with that we are pushing and popping out the elements.  
Output:

$ g++ sca2.cpp

$ a.out

100 40 30 25

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

1. #include <iostream>
2. #include <stack>
3. using namespace std;
4. int main ()
5. {
6. stack<int> myints;
7. cout << (int) myints.size();
8. for (int i = 0; i < 5; i++) myints.push(i);
9. cout << (int) myints.size() << endl;
10. return 0;
11. }

a) 05  
b) 15  
c) 24  
d) 102

Answer: a  
Explanation: In this program, We declared myints and not initialized in first option, So it’s value is 0 and on another, We are pushing 5 values, So it’s size is 5.  
Output:

$ g++ sca3.cpp

$ a.out

05

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

1. #include <iostream>
2. #include <stack>
3. using namespace std;
4. int main ()
5. {
6. stack<int> mystack;
7. mystack.push(10);
8. mystack.push(20);
9. mystack.top() -= 5;
10. cout << mystack.top() << endl;
11. return 0;
12. }

a) 10  
b) 20  
c) 13  
d) 15

Answer: d  
Explanation: In this program, We used top option and this will return the reference to the next element.  
Output:

$ g++ sca4.cpp

$ a.out

15

9. In which context does the stack operates?  
a) FIFO  
b) LIFO  
c) Both FIFO & LIFO  
d) LIFI

Answer: b  
Explanation: A stack is a container where elements operate in a LIFO context, where elements are inserted (pushed) and removed (popped) from the end of the container.

10. Which operator is used in priority queue?  
a) operator<  
b) operator>  
c) operator)  
d) operator!

Answer: a  
Explanation: It is used to decide the priority of two elements to be inserted in the queue.

# C++ Programming MCQ – Associative Containers

1. What do associate containers implement?  
a) Arrays  
b) Associative arrays  
c) Functional Arrays  
d) Static arrays

Answer: b  
Explanation: Associative containers refer to a group of class templates in the standard library of the C++ programming language that implements ordered associative arrays.

2. By using which of the following the elements in the associate container can be efficiently accessed?  
a) Key  
b) Position  
c) Both Key & Position  
d) Value

Answer: a  
Explanation: Associative containers are designed to be especially efficient in accessing its elements by their key, as opposed to sequence containers which are more efficient in accessing elements by their position.

3. How many items are presented in the associate container?  
a) 2  
b) 3  
c) 4  
d) 5

Answer: c  
Explanation: There are 4 items presented in the associate container. They are set, multiset, map and multimap.

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

1. #include <iostream>
2. #include <string>
3. #include <bitset>
4. using namespace std;
5. int main ()
6. {
7. string mystring;
8. bitset<4> mybits;
9. mybits.set();
10. mystring = mybits.to\_string<char, char\_traits<char>,
11. allocator<char> >();
12. cout << mystring << endl;
13. return 0;
14. }

a) 0000  
b) 0001  
c) 0011  
d) 1111

Answer: d  
Explanation: In this program, We converted the bitset values to string and printing it.  
Output:

$ g++ asc.cpp

$ a.out

1111

5. What will be the output of the following C++ code.

1. #include <iostream>
2. #include <algorithm>
3. #include <vector>
4. using namespace std;
5. int main ()
6. {
7. vector<int> first (5, 10);
8. vector<int> second (5, 33);
9. vector<int>::iterator it;
10. swap\_ranges(first.begin() + 1, first.end() - 1, second.begin());
11. cout << " first contains:";
12. for (it = first.begin(); it != first.end(); ++it)
13. cout << " " << \*it;
14. cout << "**\n**second contains:";
15. for (it = second.begin(); it != second.end(); ++it)
16. cout << " " << \*it;
17. return 0;
18. }

a) first contains: 10 33 33 33 10

second contains: 10 10 10 33 33

b) first contains: 10 33 33 33 10

second contains: 10 10 10 33 10

c)first contains: 10 33 33 33 30

second contains: 10 10 10 33 10

d) first contains: 10 10 10 33 30

second contains: 10 10 10 10 10

Answer: a  
Explanation: In this program, We swapped the values according to their position.  
Output:

$ g++ asc1.cpp

$ a.out

first contains: 10 33 33 33 10

second contains: 10 10 10 33 33

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

1. #include <iostream>
2. #include <map>
3. using namespace std;
4. int main ()
5. {
6. map<char, int> mymap;
7. map<char, int> :: iterator it;
8. mymap['b'] = 100;
9. mymap['a'] = 200;
10. mymap['c'] = 300;
11. for (map<char, int> :: iterator it = mymap.begin(); it != mymap.end(); ++it)
12. cout << it -> first << " => " << it -> second << '**\n**';
13. return 0;
14. }

a) a => 200

c => 300

b)a => 200

b => 100

c) a => 200

b => 100

c => 300

d) a => 200

Answer: c  
Explanation: In this program, We used the map template and then we used the begin operation and then we are printing the elements.  
Output:

$ g++ asc2.cpp

$ a.out

a => 200

b => 100

c => 300

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

1. #include <iostream>
2. #include <set>
3. using namespace std;
4. int main ()
5. {
6. set<int> myset;
7. myset.insert(20);
8. myset.insert(30);
9. myset.insert(10);
10. while (!myset.empty())
11. {
12. cout << ' ' << \*myset.begin();
13. myset.erase(myset.begin());
14. }
15. cout << '**\n**';
16. return 0;
17. }

a) 10  
b) 20  
c) 30  
d) 10 20 30

Answer: d  
Explanation: In this program, We used the set template and then we are initializing the values and then we are erasing it.  
Output:

$ g++ asc3.cpp

$ a.out

10 20 30

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

1. #include <iostream>
2. #include <set>
3. using namespace std;
4. int main ()
5. {
6. multiset<int> mymultiset;
7. for (int i = 0; i < 5; i++) mymultiset.insert(i);
8. multiset<int> :: key\_compare mycomp = mymultiset.key\_comp();
9. int highest = \*mymultiset.rbegin();
10. multiset<int> :: iterator it = mymultiset.begin();
11. do
12. {
13. cout << ' ' << \*it;
14. } while (mycomp(\*it++, highest));
15. return 0;
16. }

a) 12345  
b) 01234  
c) 1234  
d) 0123

Answer: b  
Explanation: In this program, We used the set template and then we compared the keys and printing the result.  
Output:

$ g++ asc4.cpp

$ a.out

0 1 2 3 4

9. How many instances are allowed by map and set while inserting an element into container?  
a) 1  
b) 2  
c) 3  
d) Multiple

Answer: a  
Explanation: Both map and set only allow one instance of a key or element to be inserted into the container.

10. What do maps and sets support?  
a) Single directional iterators  
b) Bi-directional iterators  
c) Single & Bi-directional directional iterators  
d) functional iterators

Answer: b  
Explanation: Bi-directional iterator are used to move in both direction from any element i.e. both forward and backward movements are allowed.

# C++ Programming Questions and Answers – Almost Containers

1. What kind of library is Standard Template Library?  
a) Polymorphic  
b) Generic  
c) Both Polymorphic & Generic  
d) Virtual

Answer: b  
Explanation: The STL is a generic library, meaning that its components are heavily parameterized.

2. To what type of object does the container can be instantiated?  
a) int  
b) float  
c) double  
d) any type of object

Answer: d  
Explanation: All type of object does the container can be instantiated.

3. What type of class template is list?  
a) Class-based  
b) Node-based  
c) Method-based  
d) size-based

Answer: b  
Explanation: It is node-based because it allows for efficient insertion anywhere in the program.

4. What type of access does deque and vector provide?  
a) Linear access  
b) Parallel access  
c) Random access  
d) Memory access  
View Answer

Answer: c  
Explanation: Because they can manipulate the values on anywhere in the program, So it is providing random access.

5. Where does the vector add the item?  
a) End  
b) Insert  
c) Middle  
d) Start

Answer: a  
Explanation: Vector allows insertion of element at the end.

6. Which are not full container classes in c++?  
a) Sequence container  
b) Associative container  
c) Container adaptor  
d) iterative container

Answer: c  
Explanation: Container adaptors are not full container classes, but classes that provide a specific interface relying on an object of one of the container classes such as deque or list to handle the elements.

7. What is the lifetime of the element in container?  
a) Whole program  
b) Outside the block  
c) Everywhere  
d) Only on that container

Answer: d  
Explanation: A Container “owns” its elements: the lifetime of an element stored in a container cannot exceed that of the Container itself.

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

1. #include <iostream>
2. #include <map>
3. using namespace std;
4. int main ()
5. {
6. multimap<char, int> mymultimap;
7. mymultimap.insert(make\_pair('x', 100));
8. mymultimap.insert(make\_pair('y', 200));
9. mymultimap.insert(make\_pair('y', 350));
10. mymultimap.insert(make\_pair('z', 500));
11. cout << mymultimap.size() << '**\n**';
12. return 0;
13. }

a) 1  
b) 2  
c) 4  
d) 3

Answer: c  
Explanation: In this program, We are counting the number of elements in the map.  
Output:

$ g++ alc.cpp

$ a.out

4

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

1. #include <iostream>
2. #include <queue>
3. using namespace std;
4. int main ()
5. {
6. priority\_queue<int> mypq;
7. mypq.push(10);
8. mypq.push(20);
9. mypq.push(15);
10. cout << mypq.top() << endl;
11. return 0;
12. }

a) 15  
b) 20  
c) 10  
d) Error

Answer: b  
Explanation: In this program, We used the queue template and the top method is used to retain the last but before element.  
Output:

$ g++ alc1.cpp

$ a.out

20

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

1. #include <iostream>
2. #include <map>
3. using namespace std;
4. int main ()
5. {
6. multimap<char, int> mymultimap;
7. mymultimap.insert(make\_pair('y', 202));
8. mymultimap.insert(make\_pair('y', 252));
9. pair<char, int> highest = \*mymultimap.rbegin();
10. multimap<char, int> :: iterator it = mymultimap.begin();
11. do
12. {
13. cout << (\*it).first << " => " << (\*it).second << '**\n**';
14. } while ( mymultimap.value\_comp()(\*it++, highest) );
15. return 0;
16. }

a) y => 202  
b) y => 252  
c) y => 202 & y => 252  
d) y => 205

Answer: a  
Explanation: In this program, the method rbegin is used to return the first element in the map.  
Output:

$ g++ alc2.cpp

$ a.out

y = &gt; 202

C++ Programming Questions and Answers – Defining a New Container

1. What do all STL containers define?  
a) Iterator types  
b) Begin methods  
c) End methods  
d) All of the mentioned

Answer: d  
Explanation: All the STL containers define the iterator types for that container, e.g., iterator and const\_iterator, e.g., vector::iterator and the begin/end methods for that container, e.g., begin() and end().

2. What do we return if we use simple array on a internal container?  
a) Methods  
b) Pointers  
c) Objects  
d) Values

Answer: b  
Explanation: Pointers are legal iterators, so if your internal container is a simple C array, then all you need to do is return the pointers.

3. What is mandatory for designing a new container?  
a) Classes  
b) Iterators  
c) Container  
d) Variables

Answer: b  
Explanation: Iterators are used to increase the generality of an algorithm. Otherwise, we need to define the algorithm for each types.

4. What are the design requirements for building a container from the sratch?  
a) Container interface requirements  
b) Allocator interface requirements  
c) Iterator requirements  
d) All of the mentioned

Answer: d  
Explanation: These are the design specific requirements for building a container from the scratch.

5. How many iterators are needed for the defining a new container?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three main iterators needed for designing a container. They are const iterator, Reverse iterator and Iterator traits.

6. What is the use of the allocator interface in the user-defined container?  
a) Storage management  
b) Memory management  
c) Storage & Memory management  
d) Iterator management

Answer: a  
Explanation: Storage management is the use of the allocator interface in the user-defined container.

7. How many types of container classes are there in c++?  
a) 1  
b) 2  
c) 3  
d) As many as possible

Answer: b  
Explanation: There are two type of container classes in c++. They are value containers and reference containers.

8. What is the name of the container which contains group of multiple objects?  
a) Heterogeneous container  
b) Homogeneous container  
c) Both Homogeneous & Heterogeneous container  
d) Sequence container

Answer: a  
Explanation: Heterogeneous container is the name of the container which contains group of multiple objects.

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

1. #include <iostream>
2. #include <string>
3. #include <algorithm>
4. using namespace std;
5. int main()
6. {
7. string s = "spaces in text";
8. s.erase(remove(s.begin(), s.end(), ' ' ), s.end() ) ;
9. cout << s << endl;
10. }

a) spaces  
b) spaces in  
c) spaces in text  
d) spacesintext

Answer: d  
Explanation: In this program, We formed a algorithm to remove spaces in the string.  
Output:

$ g++ dan.cpp

$ a.out

spacesintext

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

1. #include <vector>
2. #include <algorithm>
3. #include <iostream>
4. #include <iterator>
5. using namespace std;
6. int square(int i) { return i \* i; }
7. int main()
8. {
9. vector<int> V, V2;
10. V.push\_back(0);
11. V.push\_back(1);
12. V.push\_back(2);
13. transform(V.begin(), V.end(), back\_inserter(V2), square);
14. copy(V2.begin(), V2.end(), ostream\_iterator<int>(cout, " "));
15. cout << endl;
16. }

a) 0  
b) 1  
c) 2  
d) 0 1 4

Answer: d  
Explanation: In this program, We formed an algorithm to find the square of the given number.  
Output:

$ g++ dan1.cpp

$ a.out

0 1 4

# C++ Programming Questions and Answers – seq\_con Array Class – 1

1. What is sequence container arrays?  
a) C-like arrays  
b) Template class sequence container, alternative for C-like arrays  
c) Collection of data of the same type  
d) Collection of objects

Answer: b  
Explanation: Sequence Containers arrays are an alternative for C-like arrays. It is a static continuous array that uses template classes with extended features for array implementation.

2. Pick the correct statement.  
a) Sequence Container arrays know (somehow stores within) its size whereas C-like arrays do not  
b) Sequence Container arrays have no advantage over C-like arrays  
c) Sequence Container arrays are same as C-like arrays  
d) Sequence Container arrays are also present in C

Answer: a  
Explanation: Sequence Containers Arrays stores its size within itself so need to pass extra size parameter when passing this array as an argument.

3. Which of the following is/are advantage(s) of Sequence Container arrays over C-like arrays?  
a) Sequence Container arrays store its size within itself whereas C-like arrays do not  
b) Sequence Container arrays are more efficient  
c) Sequence Container arrays have no array decay problem whereas C-like arrays do have  
d) All of the mentioned

Answer: d  
Explanation: Sequence Container arrays(a.k.a Array classes) somehow stores its size and it can be implemented efficiently. Also, Array classes do not have Array decay problem.

4. Which of the follwoing function(s) of Array classes are similar to [] operator?  
a) at()  
b) get()  
c) both at() and get()  
d) front()

Answer: c  
Explanation: Both at() and get() function are used to access the elements stored at i’th position of the array.

5. How many different ways are there to access an element of array classes at the ith position?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three ways of accessing Array classes as mentioned below:  
i. using [] operator(same as C-like arrays)  
ii. using at() function available in array classes.  
iii. using get() function not a member of the array class.

6. What header file is included to use array classes?  
a) <array>  
b) <Array>  
c) <algorithm>  
d) <ARRAY>

Answer: a  
Explanation: <array> header file is provided by the C++ to use array classes.

7. What is the correct syntax of declaring an array class?  
a) array<type> arr;  
b) array<type,size> arr;  
c) Array<type> arr;  
d) Array<type,size> arr;

Answer: b  
Explanation: The declaration of array class starts with a keyword array followed by <> specifying the type and size of array and then the name of the identifier. Example: array<int, 10> arr; arr is an array class of type in with size = 10.

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

#include <iostream>

#include <array>

using namespace std;

int main(int argc, char const \*argv[])

{

array<int,5> arr = {1,2,3,4,5};

cout<<"Printing Using [] operator: ";

for(int i=0;i<5;i++){

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

}

cout<<endl;

cout<<"Printing Using at() function: ";

for(int i=0;i<5;i++){

cout<<arr.at(i)<<" ";

}

cout<<endl;

return 0;

}

a)1 2 3 4 5

1 2 3 4 5

b)Printing Using [] operator: 1 2 3 4 5

Printing Using at() function: 1 2 3 4 5

c)Printing Using at() function: 1 2 3 4 5

Printing Using [] operator: 1 2 3 4 5

d) Printing Using at() function: 1 2 3 4 5

Answer: b  
Explanation: In this program we are trying to print the array first using [] operator then using the at() function of the array class.  
Output:

$ ./a.out

Printing Using [] operator: 1 2 3 4 5

Printing Using at() function: 1 2 3 4 5

9. What is the syntax of printing the first element of an array Arr using get() function?  
a) Arr.get(0)  
b) get<0>(Arr)  
c) Arr.get[0]  
d) get<0>[Arr]

Answer: b  
Explanation: To access the first element of an array class Arr using get() function, we use the following get<index>(Arr) where index is an integer constant number, not an identifier.

10. Which header file is required to use get() function?  
a) <array>  
b) <tuple>  
c) <Array>  
d) <access>

Answer: b  
Explanation: <tuple> header file is required to use the get() function for accessing an element.

11. What is the difference between get() and at()?  
a) at() is available under <array> header file whereas get() is available under <tuple> header file  
b) at() is a member function of array class whereas get() is not  
c) get() takes array class as a parameter whereas at() takes a constant integer(i.e. index) as a parameter  
d) all of the mentioned

Answer: d  
Explanation: get() and at() differ in various ways. get() is not a part of array class, get is available under <tuple> header and get() takes array class also as a parameter to access the element.

12. Which function is used to access the first element of an array class?  
a) front()  
b) start()  
c) back()  
d) first()

Answer: a  
Explanation: Array class provides front() function to access the first element of the array class.

13. Which function is used to access the last element of an array class?  
a) end()  
b) start()  
c) back()  
d) last()

Answer: c  
Explanation: Array class provides back() function to access the last element of the array class.

14. Which of the following function(s) is/are used to get the size of the array class?  
a) size()  
b) max\_size()  
c) both size() and max\_size()  
d) get\_size()

Answer: c  
Explanation: Both size() and max\_size() are used to get the size of array class. There is no difference between size() and max\_size() of array class.

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

#include <iostream>

#include <array>

using namespace std;

int main(int argc, char const \*argv[])

{

array<int,10> arr = {1,2,3,4,5};

cout<<"size:"<<arr.size()<<endl;

cout<<"maxsize:"<<arr.max\_size()<<endl;

return 0;

}

a)size:10

maxsize:10

b)size:5

maxsize:10

c)size:5

maxsize:5

d)size:10

maxsize:5

Answer: a  
Explanation: Both size() and max\_size() returns the same value i.e. the size of array defined during declaration. Therefore both prints the value 10.  
Output:

$ ./a.out

size:10

maxsize:10

# C++ Programming Questions and Answers – seq\_con Array Class – 2

1. What is the use of swap() function in array class?  
a) Swaps two elements of an array given elements  
b) Swaps two arrays  
c) Swaps two elements given indices of elements  
d) Swaps same elements of the array if required

Answer: b  
Explanation: swap() function is used to swap elements of two array classes provided the size of both arrays classes are same.

2. What is the syntax of swap()?  
a) swap(arr1, arr2);  
b) arr1.swap(arr2);  
c) swap<int, int>(arr1, arr2);  
d) swap[arr1, arr2];

Answer: b  
Explanation: The correct syntax of swap function is arr1.swap(arr2) i.e. one array calling swap() function with second array as parameter to swap function. Also swap is a function therefore [] operator cannot be used to call swap function.

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

#include <iostream>

#include <array>

using namespace std;

int main(int argc, char const \*argv[])

{

array<int, 5> arr1 = {1,2,3,4,5};

array<int, 5> arr2 = {6,7,8,9,10};

arr1.swap(arr2);

for(int i=0;i<5;i++)

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

cout<<endl;

for(int i=0;i<5;i++)

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

cout<<endl;

return 0;

}

a)6 7 8 9 10

1 2 3 4 5

b)1 2 3 4 5

6 7 8 9 10

c)6 7 8 9 10

6 7 8 9 10

d)1 2 3 4 5

1 2 3 4 5

Answer: a  
Explanation: arr1 has elements from 1-5 and arr2 has elements 6-10 initially. After swapping arr1 has elements from 6-10 and arr2 has elements from 1-5. Therefore output is 6 7 8 9 10 then 1 2 3 4 5.  
Output:

$ ./a.out

6 7 8 9 10

1 2 3 4 5

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

#include <iostream>

#include <array>

using namespace std;

int main(int argc, char const \*argv[])

{

array<int, 10> arr1 = {1,2,3,4,5};

array<int, 5> arr2 = {6,7,8,9,10};

arr1.swap(arr2);

for(int i=0;i<5;i++)

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

cout<<endl;

for(int i=0;i<5;i++)

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

cout<<endl;

return 0;

}

a)6 7 8 9 10

1 2 3 4 5

b)1 2 3 4 5

6 7 8 9 10

c) Error  
d) Segmentation fault

Answer: c  
Explanation: As the size of both the array classes is not equal therefore the swap function gives an error stating that no matching function available.

5. What is the use of empty() function in array classes?  
a) To check whether the size of an array is zero or not  
b) To check whether an array is empty or not  
c) To check how many elements are there in the array  
d) To check whether an array contains negative elements or not

Answer: a  
Explanation: empty() function is used to check whether the size of an array class is zero or not. It is not used to check whether an array is empty or not. The function true only if size/max\_size of an array is zero otherwise it returns false.

6. What is the use of fill() function in array class?  
a) To fill an array with a given single value  
b) To delete all the elements that are equal to the given value  
c) To replace all the elements of the array which are equal to the given value  
d) To check whether given element fills the array or not

Answer: a  
Explanation: fill() function is used to fill an array class with the given single value.

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

#include <iostream>

#include <array>

using namespace std;

int main(int argc, char const \*argv[])

{

array<int, 5> arr1;

arr1.fill(2);

for(int i=0;i<5;i++)

cout<<arr1[i];

cout<<endl;

return 0;

}

a) 22222  
b) 20000  
c) 00002  
d) 20002

Answer: a  
Explanation: fill() function sets the value of each element equal to the value passed as parameter to the function.

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

#include <iostream>

#include <array>

using namespace std;

int main(int argc, char const \*argv[])

{

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

int arr2[5] = {6,7,8,9,10};

arr1.swap(arr2);

for(int i=0;i<5;i++)

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

cout<<endl;

for(int i=0;i<5;i++)

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

cout<<endl;

return 0;

}

a)6 7 8 9 10

1 2 3 4 5

b)1 2 3 4 5

6 7 8 9 10

c) Error  
d) Segmentation fault

Answer: c  
Explanation: swap() function is used for swapping two array classes not two C-like arrays. Therefore the swap() function gives error.

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

#include <iostream>

#include <array>

using namespace std;

int main(int argc, char const \*argv[])

{

array<int,5> arr1;

arr1.fill(5);

cout<<get<5>(arr1);

return 0;

}

a) 5  
b) Compile-time error  
c) Run-time error  
d) Segmentation fault

Answer: b  
Explanation: The compiler detects that the array class size is 5 and we are trying to access the 5th index which is out of bound therefore the program gives error.

10. What happens when both of the following C++ programs are compiled and executed?

===== Program 1 =====

#include <iostream>

#include <array>

using namespace std;

int main()

{

array<int,5> arr1;

arr1.fill(5);

cout<<get<5>(arr1);

return 0;

}

=====================

===== Program 2 =====

#include <iostream>

#include <array>

using namespace std;

int main()

{

array<int,5> arr1;

arr1.fill(5);

cout<<arr1.at(5);

return 0;

}

=====================

a) Program 1 gives compile-time error and Program 2 gives run-time error  
b) Program 1 gives run-time error and Program 2 gives compile-time error  
c) Both programs results into compile-time error  
d) Both programs results into run-time error

Answer: a  
Explanation: The Program 1 gives compile-time error whereas Program 2 gives run-time error. This is because get() function takes constant integer as the argument for accessing element of the array, therefore at compile time only the compiler verifies whether the index is accessible or not as we know the array class size during compile time, Whereas in case of at() function it takes variable as the parameter for accessing element, therefore the index range is checked during run-time therefore the error is detected during run-time.

# C++ Programming Questions and Answers – seq\_con Vector Class – 1

1. What are the vectors?  
a) Arrays with dynamic size  
b) Arrays with different types of elements  
c) Same as array classes  
d) Arrays with static size but use template classes

Answer: a  
Explanation: Vectors are just like arrays with the ability to resize itself whenever an element is added to or deleted from it.

2. Pick the correct statement.  
a) Vectors have dynamic size whereas Array classes have a static size  
b) Both vectors and Array classes have a dynamic size  
c) Both vectors and Array classes have a static size  
d) Vectors have static size whereas Array classes have a dynamic size

Answer: a  
Explanation: Vectors are implemented in a way so that it can handle any number of elements at a time means the size of a vector can vary, whereas Array classes have fixed size.

3. Pick the incorrect statement.  
a) Vectors have a dynamic size  
b) Vectors are placed in contiguous storage  
c) Insertion in vectors always takes constant time  
d) Vectors insert the element at the end

Answer: c  
Explanation: Insertion in vectors are not always constant. When we are inserting an element at the end of the vector then if a vector is full then it needs to size itself which takes time to resize and time to insert element else just time for inserting that element at the end. Hence the insertion time is not constant always. Vectors have a dynamic size. They are placed in contiguous memory for easy access.

4. Which of the following header file is needed to use vectors in your program?  
a) <array>  
b) <vector>  
c) <containers>  
d) <stdio>

Answer: b  
Explanation: Header file <vector> contains all the implementation of vector methods, hence we need to include this header file.

5. Which of the following(s) can be used to access the first element of a vector v?  
a) v.begin()  
b) v.cbegin()  
c) v[0]  
d) all of the mentioned

Answer: d  
Explanation: To access the first element of a vector we can use the following things:  
i) v.begin()  
ii) v.cbegin()  
iii) v[0]  
iv) v.at(0)

6. Which of the following(s) can be used to access the last element of a vector v?  
a) v.end()  
b) v.cend()  
c) both v.end() and v.cend()  
d) vectors do not have a function to access the last element

Answer: d  
Explanation: There are no function to access the last element of the vector. The end() and cend() returns the iterator to an element which is kept at the last of the vector to keep the knowledge about the end of a vector. In order to access the last element, you can first find the size and then can use v[size-1] or v.at(size – 1) to access the last element.

7. What is the difference between begin() and cbegin() in vectors?  
a) both are same  
b) begin() returns iterator to first element and cbegin() returns iterator to last element  
c) begin() returns an iterator to first element whereas cbegin() returns constant iterator to first element  
d) begin() returns returns first element cbegin() returns void

Answer: c  
Explanation: Both begin() and cbegin() are used to access the first element of the vector. The function begin() returns an iterator to first element whereas cbegin() returns a constant iterator to first element.

8. What is the difference between begin() and rbegin()?  
a) both are the same  
b) begin() returns an iterator to the first element and rbegin() returns an iterator to an element kept at the end of the vector  
c) begin() returns an iterator to first element whereas rbegin() returns constant iterator to first element  
d) begin() returns returns first element rbegin() returns void

Answer: b  
Explanation: begin() is used to return the iterator to the first element of the vector whereas rbegin() is used to return the an element stored at in the last of a vector.

9. Which is the following is syntactically correct for vector<int> v?  
a) vector <int> :: const\_iterator itr = v.rbegin();  
b) vector <int> :: reverse\_iterator itr = v.begin();  
c) vector <int> :: iterator itr = v.begin();  
d) vector <int> :: iterator itr = v.cbegin();

Answer: c  
Explanation: v.rbegin() returns itertor of reverse iterator therefore cannot be stored in const\_iterator(type mismatch). Similarly v.begin() returns normal iterator therefore cannot be stored in reverse\_iterator and v.cbegin() returns the const\_iterator therefore cannot be stored in normal iterator.

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

vector<int> :: const\_iterator i;

for (i = v.begin(); i != v.end(); ++i)

cout << \*i << " ";

cout<<endl;

return 0;

}

a) 1 2 3 4 5  
b) 1 3 5  
c) 1 4 5  
d) Error

Answer: a  
Explanation: A normal iterator can be stored in const\_iterator therefore program does not gives any error hence will be executed perfectly.  
Output:

$ ./a.out

1 2 3 4 5

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

vector<int> :: iterator i;

i = v.begin();

\*i = 3;

for (i = v.begin(); i != v.end(); ++i)

cout << \*i << " ";

cout<<endl;

return 0;

}

a) 1 2 3 4 5  
b) 3 2 3 4 5  
c) 5 4 3 2 1  
d) 3 3 3 3 3

Answer: b  
Explanation: We have changed the value of 0th element of vector from 1 to 3 therefore the output is as follows.  
Output:

$ ./a.out

3 2 3 4 5

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

vector<int> :: const\_iterator i;

i = v.begin();

\*i = 3;

for (i = v.begin(); i != v.end(); ++i)

cout << \*i << " ";

cout<<endl;

return 0;

}

a) 1 2 3 4 5  
b) 3 2 3 4 5  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: As i is a constant iterator therefore value stored in it is read-only therefore cannot be updated. Therefore the program gives an error.

13. Which of the following function is used to get the actual number of elements stored in vector?  
a) v.size()  
b) v.capacity()  
c) v.max\_size()  
d) v.no\_of\_elements()

Answer: a  
Explanation: To get the number of elements stored in the vector v we use the function v.size(). It returns how many elements are currently in the vector excluding the void places.

14. Which function is used to get the total capacity of a vector?  
a) v.size()  
b) v.capacity()

c) v.max\_size()  
d) v.no\_of\_elements()

Answer: b  
Explanation: capacity() function is used to get the total number of elements that can be stored at present in the vector.

15. How the size of a vector increases once it is full?  
a) Vector increases its capacity one by one  
b) Vector doubles its capacity after it is full  
c) Vector increases its capacity by half of its previous size  
d) Vector increases its capacity by a constant factor

Answer: b  
Explanation: Once the vector is full i.e. number of elements in the vector becomes equal to the capacity of the vector then vector doubles its capacity i.e. if previous capacity was 2 then new capacity becomes 2 \* 2 = 4 or 2 + 2 = 4.

**C++ Programming Questions and Answers – seq\_con Vector Class – 2**

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

cout<<v.size();

cout<<endl<<v.capacity();

return 0;

}

a)5

5

b)5

8

c)5

10

d)8

8

Answer: b  
Explanation: The size() returns the number of elements in the vector and capacity() returns the total number of elements that this vector can hold. Hence as the number of elements in vector is 5 and size is increased by 2 times. Therefore output is 5 and 8  
Output:

$ ./a.out

5

8

2. Which function is used to check whether the vector is empty or not?  
a) empty()  
b) isempty()  
c) haveElements()  
d) none()

Answer: a  
Explanation: empty() function is provided by the vector container to check whether it is empty or not.

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

v.resize(4);

for (auto it = v.begin(); it != v.end(); it++)

cout << \*it << " ";

return 0;

}

a) 1 2 3 4 5  
b) 1 2 3 4  
c) 2 3 4 5  
d) error

Answer: b  
Explanation: resize() function is used to resize a vector container. It updates the size of vector and removes all the elements after n if new size(n) is less than previous size. Hence in the program initially the vector has 5 elements but after resizing the vector to 4 it has only 4 elements as 5 is removed.  
Output:

$ ./a.out

1 2 3 4

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

cout<<v.size()<<endl;

v.resize(4);

cout<<v.size()<<endl;

return 0;

}

a)5

4

b)5

5

c)4

4

d) Error

Answer: a  
Explanation: Intitally the size of the vector is 5 as it contains only 5 elements. After resizing the elements 5 is terminated so only 4 remains therfore the size becomes 4. Hence out is as follow.  
Output:

$ ./a.out

5

4

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

cout<<v.capacity()<<endl;

v.resize(4);

cout<<v.capacity()<<endl;

return 0;

}

a)5

4

b)8

8

c)5

8

d)4

8

Answer: b  
Explanation: The capacity denotes how many elements a avector can hold. On resizing a vector the capacity of a vector is not changed hence the capacity before and after is same. Therefore the output is as follows.  
Output:

$ ./a.out

8

8

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

cout<<v.capacity()<<endl;

v.shrink\_to\_fit();

cout<<v.capacity()<<endl;

return 0;

}

a)5

8

b)8

5

c) Error  
d) Segmentation fault

Answer: b  
Explanation: Initially we have 5 elements in the vector therefore the capacity of the vector is 8(one can observe that as capacity doubles after vector is full). Now the function shrink\_to\_fit() makes the capacity of vector equal to its size hence removing the extra space occupied by the vector. Therefore as only 5 elements were there in the vectore therefore the capacity becomes 8.  
Output:

$ ./a.out

8

5

7. What will be the capacity of the vector after 10 is pushed into the vector in the following C++ code?

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

cout<<v.capacity()<<endl;

v.shrink\_to\_fit();

cout<<v.capacity()<<endl;

v.push\_back(10);

return 0;

}

a) 8  
b) 10  
c) 5  
d) 6

Answer: b  
Explanation: After shrinking the capacity of vector the capacity of vector becomes 5. Now when a new element i.e. 10 is inserted into the vector then the capacity of the vector will double i.e. it will become 10. hence the final capacity will be 10.

8. What will be the capacity of vector at the end in the following C++ code?

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

v.reserve(50);

cout<<v.capacity();

return 0;

}

a) 10  
b) 8  
c) 50  
d) 60

Answer: c  
Explanation: In this program reserve(n) function is used which is used to reserve the space for n elements in vector. Hence when the reserve(50) function is called for vector v then the we are trying to reserve memory for 50 elements, hence the capacity of vector v becomes 50.

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

int \*pos = v.data();

cout<<\*(pos + 3);

return 0;

}

a) 5  
b) 1  
c) 4  
d) 3

Answer: c  
Explanation: data() function in vector returns the direct pointer to the memory array which the vector has used to store its elements. Hence a pointer to vector is returned. So when we are accessing \*(pos + 3) we are trying to do v[3] which is 4. Hence the output is as follows.  
Output:

$ ./a.out

4

10. Which of the following function is used to insert an element at the end of a vector?  
a) push\_back()  
b) pop\_back()  
c) front()  
d) end()

Answer:a  
Explanation: Vector provides push\_back() function to insert an element at its end.

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

cout<<v.capacity()<<endl;

v.pop\_back();

v.pop\_back();

cout<<v.capacity()<<endl;

return 0;

}

a)8

4

b)4

4

c)8

8

d)4

8

Answer: c  
Explanation: Vector never reduces its capacity on deleting any element which one may get confuse by thinking about the fact that vector doubles its memory on insertion. Hence both returns the same capacity.  
Output:

$ ./a.out

8

8

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v;

for (int i = 1; i <= 5; i++)

v.push\_back(i);

for(int i=0;i<v.size();i++)

cout<<v[i]<<" ";

cout<<endl;

v.assign(3, 8);

for(int i=0;i<v.size();i++)

cout<<v[i]<<" ";

cout<<endl;

return 0;

}

a)1 2 3 4 5

8 8 8

b)1 2 3 4 5

8 8 8 8 8

c)1 2 3 4 5

8 8 8 4 5

d) 1 2 3 4 5

Answer: a  
Explanation: assign(m,n) function changes the vector values by assigning new values to vector. It copies m times the value n to the vector by first removing all the initial values. hence the vector has 3 8’s after updation i.e. using assign(3,8) function. Hence the output is as follows.  
Output:

$ ./a.out

1 2 3 4 5

8 8 8

13. Which function is used to swap two vectors?  
a) swap()  
b) change()  
c) merge()  
d) exchange()

Answer: a  
Explanation: Vectors allows the use of swap function to swap to vectors with each other of same type and size.

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v1;

vector<char> v2;

for (int i = 1; i <= 5; i++)

v1.push\_back(i);

for (int i = 6; i <= 10; i++)

v2.push\_back(i);

v1.swap(v2);

for(int i=0;i<v1.size();i++)

cout<<v1[i]<<" ";

for(int i=0;i<v2.size();i++)

cout<<v2[i]<<" ";

cout<<endl;

return 0;

}

a) 1 2 3 4 5 6 7 8 9 10  
b) 6 7 8 9 10 1 2 3 4 5  
c) Error  
d) Segmentation fault

Answer: c  
Explanation: swap() function in vector allows to swap two vectors of same size and type but here the vectors v1 and v2 have different types therefore the program gives the error.

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

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v1;

vector<int> v2;

for (int i = 1; i <= 5; i++)

v1.push\_back(i);

for (int i = 6; i <= 10; i++)

v2.push\_back(i);

v1.swap(v2);

for(int i=0;i<v1.size();i++)

cout<<v1[i]<<" ";

for(int i=0;i<v2.size();i++)

cout<<v2[i]<<" ";

cout<<endl;

return 0;

}

a) 1 2 3 4 5 6 7 8 9 10  
b) 6 7 8 9 10 1 2 3 4 5  
c) Error  
d) Segmentation fault

Answer: b  
Explanation: Here swap() function is used and the type and size of both vectors v1 and v2 are same therefore they can be swapped and hence the program allows such swap. Therefore no error and program runs perfectly.  
Output:

$ ./a.out

6 7 8 9 10 1 2 3 4 5

**C++ Programming Questions and Answers – seq\_con List**

1. How many list sequence containers are provided by STL?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: b  
Explanation: There are two list sequence containers are provided by STL namely forward\_list and list.

2. Which type of list a Forward\_list sequence container implements?  
a) Singly Linked List  
b) Doubly Linked List  
c) Both type of list  
d) A simple sequence of array

Answer: a  
Explanation: Forward\_list sequence container implements a Singly Linked List.

3. Which type of list a List sequence container implements?  
a) Singly Linked List  
b) Doubly Linked List  
c) Both type of list  
d) A simple sequence of array

Answer: b  
Explanation: List sequence container implements Doubly Linked List.

4. Which of the following header file is required for forwawrd\_list?  
a) <forward\_list>  
b) <list>  
c) <f\_list>  
d) <Forward\_List>

Answer: a  
Explanation: One needs to implement <forward\_list> header file to use forward\_list in a program.

5. Which of the following(s) is/are the correct way of assigning values to a forward\_list f?  
a) f.assign({1,2,3,4,5})  
b) f.assign(10,5)  
c) both f.assign({1,2,3,4,5}) and f.assign(10,5)  
d) f.assign(1,1,1,1)

Answer: c  
Explanation: Both f.assign({1,2,3,4,5}) and f.assign(10,5) are correct way of assigning values to a forward\_list. The first assignment initializes the list with the elements 1,2,3,4 and 5 whereas the second assignment initializes the list 10 elements with value 5 i.e. 5 10 times.

6. How the list differs from vectors?  
a) Vector is contiguous whereas List is non-contiguous  
b) Insertion in the list takes constant time whereas it is not constant in vectors  
c) There is no capacity defined for list  
d) All of the mentioned

Answer: d  
Explanation: List is non-contiguous that means elements of a list are not the contiguous manner in memory. Insertion in a list is constant for because we are not increasing the size of the list anywhere which was the case of a vector. Vectors have a capacity defined whereas there is no such capacity defined for Lists.

7. What is the syntax of declaraing a forward\_list?  
a) forward\_list f;  
b) forward\_list<type> f;  
c) forward\_list f<type>;  
d) forward\_list<type,size> f;

Answer: b  
Explanation: forward\_list<type> f; is the correct syntax of declaring a forward-list.

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

#include <iostream>

#include <vector>

#include <forward\_list>

using namespace std;

int main()

{

forward\_list<int> fl1;

fl1.assign(5,10);

for (int&c : fl1)

cout << c << " ";

cout<<endl;

fl1.insert\_after(fl1.begin(), {1,2,3});

for (int&c : fl1)

cout << c << " ";

cout<<endl;

return 0;

}

a)10 10 10 10 10

10 1 2 3 10 10 10 10

b)10 2 3 10 10 10 10

10 10 10 10 10

c) Error  
d) Segmentation fault

Answer: a  
Explanation: The program is syntactically correct therefore no error and also memory are handled carefully therefore no segmentaion fault. Hence the program runs perfectly. The insert\_after() function inserts the elements provided at the position mention in the first argument.

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

#include <iostream>

#include <vector>

#include <forward\_list>

using namespace std;

int main()

{

forward\_list<int> fl1 = {1,2,3,4,5};

for (int&c : fl1)

cout << c << " ";

cout<<endl;

forward\_list<int>::iterator ptr = fl1.begin();

fl1.erase\_after(ptr);

for (int&c : fl1)

cout << c << " ";

cout<<endl;

return 0;

}

a)1 2 3 4 5

1 2 3 4 5

b)1 2 3 4 5

1 3 4 5

c)1 2 3 4 5

2 3 4 5

d)1 2 3 4 5

1

Answer: b  
Explanation: erase\_after() function is used to erase/delete the element present next to the provided element. So in the given program we provided fl1.begin() i.e. 1 as the element to erase\_after() function hence the element after 1 i.e. 2 is deleted.

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

#include <iostream>

#include <vector>

#include <forward\_list>

using namespace std;

int main()

{

forward\_list<int> fl1 = {1,2,3,4,5};

for (int&c : fl1)

cout << c << " ";

cout<<endl;

fl1.remove\_if([](int x){ return x > 3;});

for (int&c : fl1)

cout << c << " ";

cout<<endl;

return 0;

}

a)1 2 3 4 5

1 2 3 4 5

b)1 2 3 4 5

1 2

c)1 2 3 4 5

1 2 3

d)4 5

1 2 3 4 5

Answer: c  
Explanation: remove\_if() function is provided in list to remove element based on the conditions provided in the function. So in the program we asked to delete all the element which are greater then 3, hence 4 and 5 are deleted and we are remained with 1,2 and 3.  
output:

$ ./a.out

1 2 3 4 5

1 2 3

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

#include <iostream>

#include <vector>

#include <forward\_list>

using namespace std;

int main()

{

forward\_list<int> fl1 = {1,7,8,9,10};

forward\_list<int> fl2 = {2,3,4,5,6};

fl1.splice\_after(fl1.begin(), fl2);

for (int&c : fl1)

cout << c << " ";

cout<<endl;

return 0;

}

a) 1 2 3 4 5  
b) 1 2 3 4 5 6 7 8 9 10  
c) 1 7 8 9 10  
d) 2 3 4 5 6

Answer: b  
Explanation: splice\_after() function is used to insert a forward-list into another list after a given position. So in this program we are trying to insert list2 into list1 after fl1.bein() i.e. 1. Hence the list1 becomes 1 2 3 4 5 6 7 8 9 10.  
Output:

$ ./a.out

1 2 3 4 5 6 7 8 9 10

**C++ Programming Questions and Answers – STL – Pair**

1. What is a pair?  
a) Container consisting of two data elements of the same type  
b) Container consisting of two data elements of different type  
c) Container consisting of one header and two data elements of the same type  
d) Container consisting of two data elements can have the same or different type

Answer: d  
Explanation: Pair is a container defined in STL which consist of two elements which can be of same or different types.

2. Which header file is required to use pair container in your program?  
a) <algorihtm>  
b) <utility>  
c) <pair>  
d) <utitityPair>

Answer: b  
Explanation: Pair container is defined under the header file <utility> therefore one should include header before using pair container.

3. Which of the following is the correct syntax of using pair p?  
a) pair <type,type> p;  
b) pair p <type,type>;  
c) pair [type,type] p;  
d) pair p [type,type];

Answer: a  
Explanation: A pair is declared using the this syntax pair <type, type> identifier.

4. Which of the following operations can be performed on a pair?  
a) Assignment of pairs  
b) Copying of one pair to another  
c) Comparison of two pairs  
d) All of the mentioned

Answer: d  
Explanation: A pair can be assigned, copied or can be compared. Hence all the above operations can e performed on pairs.

5. Which operator is used to access the first or second element of a pair?  
a) ->  
b) .  
c) \*  
d) []

Answer: b  
Explanation: .(dot) operator is used to access the first or second element of a pair. For example, if p = (1,2) is a pair then 2 can be accessed by using p.first and 2 can be accessed using p.second.

6. Which of the following is the correct syntax of accessing the first element of a pair p?  
a) p.first  
b) p.second  
c) p[0]  
d) p[1]

Answer: a  
Explanation: To access the first element of a pair we use first. for example, if p = (1,2) is a pair then we will use p.first to access the first element of the pair.

7. Which of the following is the correct syntax of accessing the second element of a pair p?  
a) p.first  
b) p.second  
c) p[0]  
d) p[1]

Answer: b  
Explanation: To access the second element of a pair we use second. for example, if p = (1,2) is a pair then we will use p.second to access the second element of the pair.

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

#include <iostream>

#include <utility>

using namespace std;

int main ()

{

pair <int,int> p(1,2);

cout<<"Pair(first,second) = ("<<p.first<<","<<p.second<<")**\n**";

return 0;

}

a) Pair(first,second) = (1,2)  
b) Compile-time error  
c) Run-time error  
d) Assignment is not correct

Answer: a  
Explanation: This is a way of assigning a pair therefore the program is correct hence the program runs perfectly and outputs the value as follows.  
Output:

$ ./a.out

Pair(first,second) = (1,2)

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

#include <iostream>

#include <utility>

using namespace std;

int main ()

{

pair p(1,2);

cout<<"Pair(first,second) = ("<<p.first<<","<<p.second<<")**\n**";

return 0;

}

a) Pair(first,second) = (1,2)  
b) Compile-time error  
c) Run-time error  
d) Assignment is not correct

Answer: b  
Explanation: A pair always expects tempalte arguments i.e. types of first and second during declaration of pair. In this program as we have not mentioned the template arguments i.e. types of first and second therefore the program gives and error.

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

#include <iostream>

#include <utility>

using namespace std;

int main ()

{

pair <int,int>p;

p = make\_pair(1,2);

cout<<"Pair(first,second) = ("<<p.first<<","<<p.second<<")**\n**";

return 0;

}

a) Pair(first,second) = (1,2)  
b) Compile-time error  
c) Run-time error  
d) Assignment is not correct

Answer: a  
Explanation: make\_pair() is a function provied to define the values for a pair. Hence the program is correct therefore the program runs successfully.  
Output:

$ ./a.out

Pair(first,second) = (1,2)

11. Which of the following is correct way of copying the values of pair p1 into other pair p2?  
a) pair <type,type> p2 = p1;  
b) pair <type,type> p2(p1);  
c) Both pair <type,type> p2 = p1; and pair <type,type> p2(p1);  
d) Pair <int,int> p2.copy(p1);

Answer: c  
Explanation: Both pair <type,type> p2 = p1; and pair <type,type> p2(p1); can be used to copy the data of one pair into other pair.

12. What happens if a pair is not initialized?  
a) Both first and second part is initialized to zero or null  
b) Both first and second part is initialized a garbage value  
c) First is initialized to zero or null and second is initialized a garbage value  
d) Second is initialized to zero or null and first is initialized a garbage value

Answer: a  
Explanation: If a pair is not initialized then by default both parts of the pair is initialized to zero.

13. Which of the following Operator cannot be used with pairs?  
a) +  
b) ==  
c) =  
d) !=

Answer: a  
Explanation: We can use only assignment and logical operators with pairs.

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

#include <iostream>

#include <utility>

#include <string>

using namespace std;

int main ()

{

pair <int,int> p1(1,2);

pair <int,int> p2(3,4);

cout<<"Pair(first,second) = ("<<p1.first<<","<<p1.second<<")**\n**";

p1.swap(p2);

cout<<"Pair(first,second) = ("<<p1.first<<","<<p1.second<<")**\n**";

return 0;

}

a)Pair(first,second) = (1,2)

Pair(first,second) = (3,4)

b)Pair(first,second) = (3,4)

Pair(first,second) = (1,2)

c)Pair(first,second) = (1,2)

Pair(first,second) = (1,2)

d)Pair(first,second) = (3,4)

Pair(first,second) = (3,4)

Answer: a  
Explanation: Inititally the pair p1 = (1,2) therefore Pair(first,second) = (1,2) is printed and when we have used swap function to swap p1 with p2 the p1 and p2 is swapped therefore next time Pair(first,second) = (3,4) is printed.  
Output:

$ ./a.out

Pair(first,second) = (1,2)

Pair(first,second) = (3,4)

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

#include <iostream>

#include <utility>

#include <string>

using namespace std;

int main ()

{

pair <int,int> p1(1,2);

pair <int,int> p2(3,4);

if(p1 <= p2)

cout<<"P1 is small";

else

cout<<"P2 is small";

return 0;

}

a) P1 is small  
b) P2 is small  
c) Error  
d) Segmentation fault

Answer: a  
Explanation: As both the elements are small in p1 pair, therefore, the pair p1 is considered small hence the output is as follows.  
Output:

$ ./a.out

P1 is small

**C++ Programming Questions and Answers – STL Container Any – 1**

1. What is any in C++?  
a) STL container used to store a single value of any type  
b) Exception class in C++  
c) Fundamental type provided by C++  
d) Template data type

Answer: a  
Explanation: Any is an STL container provided by C++ to store value or objects of any type.

2. In how many different ways any-container can be constructed?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: There are three basic ways of constructing any variable. They are done using copy initialization, using the constructor or using an assignment operator.

3. What is the correct syntax of constructing any using copy initialization?  
a) any variable\_name = value;  
b) any variable\_name(value);  
c)any variable\_name;

variable\_name = value;

d) any <type>variable\_name = value;

Answer: a  
Explanation: To initialize an any variable using copy initialization we use the following syntax:

any variable\_name = value;

4. What is the correct syntax of constructing any using parameterized constructor?  
a) any variable\_name = value;  
b) any variable\_name(value);  
c)any variable\_name;

variable\_name = value;

d) any <type>variable\_name = value;

Answer: b  
Explanation: To initialize an any variable using parameterized constructor we use the following syntax:

any variable\_name(value);

5. What is the correct syntax of constructing any using assignment operator?  
a) any variable\_name = value;  
b) any variable\_name(value);  
c)any variable\_name;

variable\_name = value;

d) any <type>variable\_name = value;

Answer: b  
Explanation: To initialize an any variable using assignment operator we use the following syntax:

any variable\_name;

variable\_name = value;

6. Which of the following syntax is used to convert any variable to its original type?  
a) any\_cast<variable\_name>();  
b) any\_cast(variable\_name);  
c) <original\_type>(variable\_name);  
d) any\_cast<original\_type>(variable\_name);

Answer: d  
Explanation: The syntax used to convert the any variable to its original type is as follows:

any\_cast(variable\_name);

7. Which header file is required to use any container?  
a) <any>  
b) <stl>  
c) <container-any>  
d) <containers>

Answer: a  
Explanation: <any> header file is required to use any container and its realted functions.

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

#include<iostream>

#include<any>

using namespace std;

int main()

{

int a = 5;

any var = a;

cout<<var<<endl;

return 0;

}

a) 5  
b) Compile-time error  
c) Run-time error  
d) Nothing is printed

Answer: b  
Explanation: C++ does not allow programmer to directly print the value of any container variable. One need type cast the any variable before printing.

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

#include<iostream>

#include<any>

#include<string>

using namespace std;

int main()

{

string s = "Hello World";

any var(s);

cout<<any\_cast<string>(var)<<endl;

return 0;

}

a) Run-time error  
b) Compile-time error  
c) Hello World  
d) Nothing is printed

Answer: d  
Explanation: In the above program as we have converted the value to its original type before printing therefore the program runs perfectly and outputs “Hello World”.

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

#include<iostream>

#include<any>

#include<string>

using namespace std;

int main()

{

string s = "Hello World";

any var(s);

cout<<any\_cast<char\*>(var)<<endl;

return 0;

}

a) Hello World  
b) Compile-time error  
c) Run-time error  
d) Nothing is printed

Answer: c  
Explanation: In this program as we are trying to convert an string into char\* which is not same therefore the program gives run-time error saying bad\_any\_cast.

**C++ Programming Questions and Answers – STL Container Any – 2**

1. Which exception is thrown if the typecasting is not done properly?  
a) bad\_type\_cast  
b) bad\_any\_cast  
c) type\_mismatched  
d) bad\_cast\_mismatched

Answer: b  
Explanation: bad\_any\_cast exception is thrown when typecasting is not done properly by the user i.e. if any is storing int value and we are trying to cast it into a string then the program will throw bad\_any\_cast exception.

2. What is the use of emplace() function?  
a) Used to change the object any container is holding  
b) Used to add more item to the any list  
c) Used to empty any container value  
d) Used to check the type of any variable

Answer: a  
Explanation: emplace() function is used to change the object contained in any container i.e destroying the present object and creating the new object for the value given by the user.

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

#include<iostream>

#include<any>

using namespace std;

int main()

{

float val = 5.5;

any var(val);

cout<<var<<endl;

char c = 'a';

var.emplace<char>(c);

cout<<var<<endl;

return 0;

}

a) 5.5  
b) a  
c)5.5

a

d) Error

Answer: c  
Explanation: In this program we are using emplace() function to change the any variable contents and this is allowed in C++ therefore the program runs fine.

4. What is the use of type() function in any container?  
a) Used to destroys the contained object in any variable  
b) Used to change the object any container is holding  
c) Used to return the type information about the any container  
d) Used to check whether a container is empty or not

Answer: c  
Explanation: type() function is used to check the type of data/value the container object is holding.

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

#include<iostream>

#include<any>

using namespace std;

int main()

{

float val = 5.5;

any var(val);

cout<<var.type().name()<<endl;

return 0;

}

a) f  
b) d  
c) Pkc  
d) u

Answer: a  
Explanation: The type function is used to get information about the data stored in the any container variable. name() attribute is used to print the type id of the data. Now as the data stored in any variable is float therefore the program outputs f as f is the type id for float.

6. What is the use of has\_value() function in any container?  
a) Used to destroys the contained object in any variable  
b) Used to change the object any container is holding  
c) Used to return the type information about the any container  
d) Used to check whether any container is empty or not

Answer: d  
Explanation: has\_value() function is provided to check whether a given any container is empty or not.

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

#include<iostream>

#include<any>

using namespace std;

int main()

{

float val = 5.5;

any var(val);

if(var.has\_value())

{

cout<<"Var is not Empty**\n**";

}

else

{

cout<<"Var is Empty**\n**";

}

return 0;

}

a) Var is Empty  
b) Var is not Empty  
c) Error  
d) Segmentation fault

Answer: b  
Explanation: As the variable is containing the information about the float value val = 5.5 therefore the container is not empty therefore the program outputs “Var is not Empty”.

8. What is the use of reset() function?  
a) Used to destroys the contained object in any variable  
b) Used to change the object any container is holding  
c) Used to empty any container value  
d) Used to check the type of any variable

Answer: a  
Explanation: reset() function is provided with any to destroy an object contained in any variable in case it is not needed.

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

#include<iostream>

#include<any>

using namespace std;

int main()

{

float val = 5.5;

any var(val);

cout<<any\_cast<float>(var)<<endl;

var.reset();

if(!var.has\_value())

{

cout<<"var is empty**\n**";

}

else{

cout<<"var is not empty**\n**";

}

return 0;

}

a) var is empty  
b)5.5

var is not empty

c) 5.5  
d)5.5

var is empty

Answer: d  
Explanation: As the program uses reset() function which resets/destroys an object contained inside the any container therefore var becomes empty hence the program outputs “var is empty”.

10. In how many ways we can handle errors in any class?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: b  
Explanation: There are two ways of handling errors in any container first by using exceptions like bad\_any\_cast and second by returning the pointer.

# 