Assignment - 36 **A Job Ready Bootcamp in C++, DSA and IOT**

STL: array

1. Using STL Array gets and sets a reference to an element based on a given index.

```
#include <iostream>
#include <array>
#include <tuple>
using namespace std;

int main()
{
    array<int, 6> a; // create array object

    cout << "Enter 6 elements: ";

    // input in array object a - elements
    for (int i = 0; i < a.size(); i++)
        cin >> a.at(i);

    cout << "Elements are: ";

    // print array object a - elements
    for (int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    // access array elemnts based on index
    cout << "\n";
    cout << "index[1] = " << get<1>(a) << " ";
    cout << "index[2] = " << get<2>(a) << " ";
    return 0;
}
```

```
Output:
Enter 6 elements: 1 2 3 4 5 6
Elements are: 1 2 3 4 5 6
index[1] = 2 index[2] = 3
```

2. Using STL Array returns the total number of elements in the array.

```
#include <iostream>
#include <array>
#include <tuple>
using namespace std;

int main()
{
    array<int, 6> a = {1, 5, 8, 9, 3, 9};
    int count = 0;
    cout << "Elements are: ";

    // print array object a - elements
    for (auto i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    // access array elemnts based on index
    for(auto i = a.begin(); i != a.end(); i++)
        count++;

    cout << endl << "The total number of elements in the array is " << count
<< endl;
    return 0;
}
```

```
Output:
Elements are: 1 5 8 9 3 9
The total number of elements in the array is 6
```

3. Find the first and last element using the STL array.

```
#include <iostream>
#include <array>
#include <tuple>
using namespace std;

int main()
{
    array<int, 5> a = {1, 5, 8, 9, 3};
    cout << "Elements are: ";

    // print array object a - elements
    for (int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    cout << "\nThe first element in array: " << a.front();
    cout << "\nThe last element in array: " << a.back();

    return 0;
}
================================================================================
Output:
Elements are: 1 5 8 9 3
The first element in array: 1
The last element in array: 3
```

4. Returns the element from the given index using the STL array.

```
#include <iostream>
#include <array>
#include <tuple>
using namespace std;

int main()
{
    array<int, 10> a = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int r =  get<5>(a);
    cout << r << endl;
    r = a[5];
    cout << r << endl;
    r = a.at(5);
    cout << r << endl;
    return 0;
}
================================================================================
Output:
6
6
6
```

5. C++ STL program to demonstrate example of array::rbegin() and array::rend() functions

```
#include <iostream>
#include <array>
#include <tuple>
using namespace std;

int main()
{
    array<int, 6> a = {1, 5, 8, 9, 3, 7};
```

```
    cout << "Elements are: ";

    // print array object a - elements
    for (int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    // print reverse order array object elements
    cout << "\n";
    for (auto i = a.rbegin(); i != a.rend(); i++)
        cout << *i << " ";

    return 0;
}
```

```
Output:
Elements are: 1 5 8 9 3 7
7 3 9 8 5 1
```

6. Using STL to check whether an array is empty or not.

```
#include <iostream>
#include <array>
#include <tuple>
using namespace std;

int main()
{
    array <int, 10> a = {1,2,3,4,5,6,7,8,9,10};

    // print all elements
    for(int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    a.fill(0);  // fill all elements as zero
    cout << endl;

    // again print array object elements
    for(int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    cout << endl;

    if(a.empty())   // check array-a empty or not
        cout << "array-a is empty" << endl;
    else
        cout << "array-a is not empty" << endl;

    array <int, 0> b; // empty array

    if(b.empty())   // check array-b empty or not
        cout << "array-b is empty" << endl;
    else
        cout << "array-b is not empty" << endl;

    return 0;
}
```

```
Output:
1 2 3 4 5 6 7 8 9 10
0 0 0 0 0 0 0 0 0 0
array-a is not empty
array-b is empty
```

7. Sort an array in ascending order using sort() function in C++ STL

```cpp
#include <iostream>
#include <array>
#include <algorithm>
using namespace std;

int main()
{
    array<int, 10> a = {2, 56, 8, 7, 6, 78, 5, 98, 25, 75};

    // print all elements
    cout << "Before sorting" << endl;
    for (int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    cout << endl;

    sort(a.begin(), a.end()); // sort function to short arrray of elements

    cout << "After sorting ascending order" << endl;
    for (int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    return 0;
}
```

```
Output:
Before sorting
2 56 8 7 6 78 5 98 25 75
After sorting ascending order
2 5 6 7 8 25 56 75 78 98
```

8. Sort an array in descending order using sort() function in C++ STL

```cpp
#include <iostream>
#include <array>
#include <algorithm>
using namespace std;

int main()
{
    array<int, 10> a = {2, 56, 8, 7, 6, 78, 5, 98, 25, 75};

    // print all elements
    cout << "Before sorting" << endl;
    for (int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    cout << endl;

    sort(a.begin(), a.end(), greater<int>()); // sort function to short
arrray of elements

    cout << "After sorting descending order" << endl;
    for (int i = 0; i < a.size(); i++)
        cout << a.at(i) << " ";

    return 0;
}
```

```
Output:
Before sorting
2 56 8 7 6 78 5 98 25 75
After sorting descending order
98 78 75 56 25 8 7 6 5 2
```

9. C++ program to find the integers which come an odd number of times in an array using C++ STL.

```
#include <iostream>
#include <array>
#include <algorithm>
using namespace std;

int oddInteger(array<int, 5> a)
{
    int result = 0;
    for(int i = 0; i < a.size(); i++)
        result = result ^ a[i];
    return result;
}
int main()
{
    array<int, 5> a = {1, 4, 4, 9, 1};
    int result = oddInteger(a);

    cout << result << endl;
    return 0;
}
================================================================================
Output:
9
```

10. Given an integer array nums , return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i] .

```
#include <iostream>
#include <array>
using namespace std;

int main()
{
    array<int, 5> num = {-1, 1, 0, -3, 3};  // initialize array object

    // print array object elements through loop
    for (int i = 0; i < num.size(); i++)
        cout << num.at(i) << " ";

    cout << endl;
    int answer = 1;

    //
    for (int i = 0; i < num.size(); i++)
    {
        answer = 1;
        for (int j = 0; j < num.size(); j++)
            if (i != j)
                answer = answer * num.at(j);

        cout << answer << ", ";
    }

    return 0;
}
================================================================================
Output:
-1 1 0 -3 3
0, 0, 9, 0, 0,
```