1. Write a factory function template **make2DArray** that returns a 2D array of elements of a given type parameter **T** filled with a specified initial value of type **T**. Your function must use **std::array** as its underlying storage, must not use loops, and must support the following code segment:

```
auto a = make2DArray<int>(99); // a is 1x1 2D array, filled with 99
auto b = make2DArray<int, 5>(88); // b is 5x1 2D array, filled with 88
auto c = make2DArray<int, 5, 10>(77); // c is 5x10 2D array, filled with 77
```

2. Using the **std::copy** algorithm, write code that echoes **char**s form the standard input stream **cin** to the standard output stream **cout**. Each input character echoed must be followed by a blank space. You code must accomplish its task without using explicit loops. Here is an sample output of the echoing process:

```
Echo this
E c h o   t h i s
 echo t h i s
e c h o   t   h   i   s
 stop it!
s t o p   i t !
 ^Z
```

Note that whitespaces are not skipped during input processing.

3. Write a function **sortVector** that accepts as input a **vector<int>** by reference and returns a sorted version of the vector. Implement the function in two different ways:

1. Use a **std::multiset<int>**. (Why use a **std::multiset** rather than a **std::set**?).
2. Use the **std::sort** algorithm.

4. Write a function **AltSum** that accepts as parameter a **vector<int>** by reference. The function function must return the alternating sum of the vector's entries. For example, if the input vector contains 5, 6, 7, 8, 9 the function would return 5 - 6 + 7 - 8 + 9 = 7. The function must not use any loops; instead, use **accumulate** and a custom functor (function object) class that performs the the addition and subtraction.

5. The STL **generate_n** algorithm is defined as

```
template <class OutputIterator, class Size, class Generator>
  void generate_n ( OutputIterator first, Size n, Generator gen )
{
  while (n>0) {
    *first = gen();
    ++first; --n;
  }
}
```

As you can see, the algorithm calls the zero-parameter function **gen n** times, storing the returned values in the range starting at **first**.

Write a function with the following prototype:

```
std::vector<int> MakeAndFillVector(int n, int x, int y);
```

The function must return a **vector<int>** filled with **n** random integers between **x** and **y**, inclusive. Do not use any loops. Instead, use the **generate_n** algorithm and a lambda expression that uses the **rand()** from the <**cstdlib**> header to generate the random integers.