ACM Programming Competition  
Worksheet 4  
Arrays, Vectors and Dequeue

1. Note, you will need a book covering the STL templates to take to the competition.

2. It is very often necessary to store an array of data. In C++ there are templates that allow this to be done very easily. For example, to store an array of integers you could add code:

   ```cpp
   #include <vector>
   using namespace std
   typedef vector<int> IntArray;
   IntArray thisArray;
   ```

3. The program `vectors.cpp` and its data file `vectors.txt`—on our web page—show an example of the simple use of a vector.

4. However a vector also has lots of extra methods that are useful, most of these rely on using an iterator into the vector. Iterators are somewhat like pointers. The code in `vectors2.cpp` shows the same program as `vectors.cpp`, but the processing is done with an iterator.

5. Methods used commonly in vector (and other templates in the STL):

<table>
<thead>
<tr>
<th>Method</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>array.clear()</td>
<td>Empties the array</td>
</tr>
<tr>
<td>itr = array.begin()</td>
<td>Points itr at the beginning of the array</td>
</tr>
<tr>
<td>itr = array.end()</td>
<td>Points itr at the ‘point’ just past the last data in the array</td>
</tr>
<tr>
<td>itr = array.find(data)</td>
<td>itr now points to the data found or to array.end()</td>
</tr>
<tr>
<td>itr = array.erase(itr)</td>
<td>Before the call: itr pointed to the data to be erased</td>
</tr>
<tr>
<td></td>
<td>After the call: itr points to the next item in the array</td>
</tr>
<tr>
<td>itr = array.insert(itr, data)</td>
<td>Inserts the new data just before the data to which itr points</td>
</tr>
<tr>
<td></td>
<td>After the call: itr points to the new data in the array</td>
</tr>
<tr>
<td>array.push_back(data)</td>
<td>Adds data to the end of the array</td>
</tr>
<tr>
<td>array.pop_back()</td>
<td>Removes the last item from the array</td>
</tr>
</tbody>
</table>

6. The STL deque class is identical to vector, except that you can push and pop at the front as well as the back of the structure.
The STL, Iterators and Algorithms

Note: when using the STL, don’t forget to include the appropriate header file as well as the <algorithm> header file.

STL vectors (etc) can be accessed in two ways, by index and by iterator. Iterators are used by the STL methods for finding, deleting, sorting etc.

Note that all STL containers have an empty() function and most have a size() function.

```cpp
// Define a vector of integers
typedef vector<int> IntVec;

// Declare a vector of integers
IntVec numbers;

// Put 100 random integers between 20 and 29 (inclusive) into the vector
for (int index = 0; index < 100; index++)
{
    int num = rand() % (30-20) + 20
    numbers.push_back(num);
}

// Output the integers using an index
int size = numbers.size();
for (int index = 0; index < size; index++)
{
    cout << numbers[index] << 
}

// Output the integers using an iterator
IntVec::iterator itr;
for (itr = numbers.begin(); itr != numbers.end(); itr++)
{
    cout << *itr << 
}

// And if you want the array sorted:
sort(numbers.begin(), numbers.end());

// Finding the first occurrence of 25 in the vector
IntVec::iterator itr;
itr = find(numbers.begin(), numbers.end(), 25);
cout << *itr;
```

cont...
// Deleting all integers after the first occurrence of 25 in the vector

IntVec::iterator itr;
itr = find(numbers.begin(), numbers.end(), 25);
numbers.erase(itr, numbers.end());

// Copying all integers after the first occurrence of 25 to the
// end of another vector

IntVec::iterator itr;
itr = find(numbers.begin(), numbers.end(), 25);
vector2.insert(vector2.end(),itr, numbers.end());

// Deleting all integers equal to 25 - method 1

IntVec::iterator itr;
for (itr = numbers.begin(); itr != numbers.end();)
{
    if (*itr == 25)
    {
        itr = numbers.erase(itr);
    }
    else
    {
        itr++;
    }
}

// Deleting all integers equal to 25 - method 2

IntVec::iterator itr;
itr = find(numbers.begin(), numbers.end(), 25);
while (itr != numbers.end())
{
    numbers.erase(itr);
    itr = find(numbers.begin(), numbers.end(), 25);
}
Task

1. A program is required that will search a series of numbers for another number (input first). If the number is found, then the program must output the series of numbers without the one found. If it is not found then the program will output `<number> not found`. It must deal with multiple sets of data.

For example, given the following data:

```
12 23 45 67 89 12 45 23 89
25 67 14 83 92 83
56 45 56 21 89 10 11 33 100 31 28
```

The output would be:

```
23 45 67 89 45 23 89\n25 not found\n45 2 89 10 11 33 100 31 28
```

2. Come up with some more test data and output: think of boundary conditions, unusual conditions etc.

3. Write a program to perform this task.