Arrays

Arrays are our first real look at data abstraction. An array is an ordered collection of data values, but it can be described as if it were a singular "thing". So, for example, a "deck" of cards might be represented using a 52-element array of characters or integers. We could subsequently pass this "deck" to a function that might "deal" the cards one by one.

Arrays or lists of data are integral to problem solving in every programming language. In this lab, we begin exploring this often-used and important concept.

Mystery-Box Challenge

Here is your next mystery-box challenge. Determine what the following function does and explain it to one of your TAs.

```cpp
bool mystery(string fstr)
{
    string rstr;
    for(int i=fstr.length()-1; i>=0 ;i--)
        rstr += fstr[i];
    // fstr[i] gets char at index i, as if it was a char[]
    return rstr == fstr;
}
```

Warm-up

1) Declaring/initializing arrays
Download partner.cpp from the website (under the lab section). Currently, there are two variables for names (fullName1 and fullName2) along with two variables for heights (height1 and height2). Rewrite the code to use a single array for each of these pairs of variables (a single array for all names and another array for all heights). The code should give the use the same input/output as before.

2) Passing arrays
Copy-paste your code from warm-up 1 into a new file as a starting point for this problem. In main() there is a cout statement. Make a new function that displays the same information as this cout and call that function in main() instead. (Hint: Cut and paste the cout statement into a function and pass the correct argument.)

Stretch

1) Partially filled arrays
Copy-paste your code from warm-up 2 into a new file as a starting point for this problem. Modify your main() function to allow any number of names and heights to be entered (instead of just 2). You will need to use a partially filled array to do this. (You can assume that there will not be more than 100 people.)
Workout

1) Concatenating arrays
Strings have build-in functionality to concatenate together. Write a function called append() that does concatenation for two char arrays (you will have to pass in a third char array that will hold the result). You will also need to pass in the lengths of each char array. You may not use strings for this part.

Sample main code:

    char first[] = {'I', ' ', 'a', 'm', ' '};
    char second[] = {'f', 'i', 'n', 'i', 's', 'h', 'e', 'd', '\0'};
    char result[200];
    append(first, 5, second, 9, result);
    cout << result;

Example output for above:
    I am finished

2) Bubble Sort
Sorting a list of numbers is an important Computer Science problem that has been extensively studied. One of the simplest methods is known as “bubble sort”. Given a list of numbers:

    3 5 2 8 9 1

the basic idea is to compare the first two numbers in the list and swap them if the first one is larger than the second (assuming you wish to sort from low to high). Next, the second and third numbers are compared and swapped if necessary, then the third and fourth, fourth and fifth, and so on until the entire list has been examined. After the first pass through, the list would look like this:

    3 2 5 8 1 9

Note that the largest value will always end up in the last position after the first pass.

Next, we repeat the process, “bubbling” the larger values up in the list on each pass. Note however, that for the second pass we don’t need to examine the last value because it’s guaranteed to be the largest. After the second pass, the last two values need not be examined, and so on.

The process ends when an entire pass through the list results in no values being swapped.

Write a program that will implement the Bubble-Sort algorithm. For this problem you need to do the following:

- Declare an integer array named list that contains 50 values.
- Using a loop, initialize list with the values 100, 99, 98, … (in decreasing order)
- Construct a void function named bsort that implements the Bubble Sort algorithm as described above. Your function should accept two arguments: the integer array to be sorted and the number of elements in the array.
- Call the bsort function to arrange the elements of list in increasing order.
- Print out the elements of list, 5 per line as follows:

Example:

<table>
<thead>
<tr>
<th>51</th>
<th>52</th>
<th>53</th>
<th>54</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
</tr>
</tbody>
</table>

(and so on...)
[Hint: This problem will be much easier if you attack it in stages. First, write a \texttt{swap} function that will swap two integer values, then write another function that will make a single pass through the array, calling the \texttt{swap} function to correct any out-of-order elements. Finally, call this second function as many times as necessary to sort the array]

**Check**
Individually, list one important things you learned in lab today, one question you still have about the lab material, and one way arrays are used in your major. When you are done, share your list with your partner.

**Challenge**
Here is an extra challenge problem. You should try to complete the warm-up, stretch and workout problems in the lab. Try this challenge problem if you have extra time or would like additional practice outside of lab.

1) **Non-repeating letters**
Write a \texttt{noRepeat()} function that takes three char arrays and two ints as arguments. The \texttt{noRepeat()} function should first concatenate two of the char arrays (using workout 1), then copy the letters into the third char array without any of the characters repeating (only the first time the character is encountered should it be in the array). After copying into the third array, add a null terminator (`\texttt{'\0'}`) at the end to ensure \texttt{cout} does not print funny things. Sample main() code is provided below. Again you may not use strings for this part.

Sample main code:

```cpp
char x[] = {'h', 'e', 'l', 'l', 'o', ' '};
char y[] = {'l', 'l', 'a', 'm', 'a'};
char result[200];
noRepeat(x, 6, y, 5, result);
cout << result;
noRepeat(y, 5, x, 6, result);
```

Example output for above:
```
heo am
lamheo
```