Hey everyone! Hope everyone is doing well! We are halfway through the semester so the pace for most classes picks up a little. Make sure to try to keep your spirits high, do small things to reward yourself for finishing assignments and such, and just try to give yourself some breaks! Just as a reminder: These documents are intended for going over concepts that are discovered in class, as they are a concise and to the point review of the materials. But remember, these documents are not a replacement for lecture and they cannot cover everything, so make sure you get help on any specific concepts you struggle with as well as go over lecture notes, programs, and the textbook.

**PLEASE NOTE:** Our group tutoring sessions will unfortunately have to be moved to Tuesdays from 6:15PM - 7:15PM, in Cashion C316. You can get more information and reserve a spot at [https://baylor.edu/tutoring](https://baylor.edu/tutoring).

*Keywords: Vectors, String and CString Functions*

**TOPIC OF THE WEEK**

**Vectors**

Last week we talked about Arrays within C++. One of the limitations of an Array is it having to have a fixed size. This means that once the array is declared, then there can not be a change to the size of an array. To combat this, we can use what is called a vector. A vector has the ability to change size as we add more elements to it. Vectors are included within the C++ Standard Template Library, the STL, and we can use vectors by using the ‘#include <vector>’ preprocessor directive.

For example, you can declare a vector of integers, named *myVector*, by doing the following declaration:

```cpp
vector<int> myVector();
```
This creates an empty vector of integers. **We are able to add numbers to the vector by using the `push_back()` function!** If we wanted to add the numbers 1 & 2 to the `myVector` vector, then we can do the following:

```cpp
define myVector.push_back(1);
define myVector.push_back(2);
```

Now we have a vector with the value, 1, in position 0...and value, 2, in position 1. Also, although we know that there are two numbers inside of the vector, there are instances, such as looping, that we would like to know the size of the vector internal to the program (rather than just knowledge). For this, we can use a `size()` function similar to that of strings, in order to find out the current size of a vector.

```cpp
myVector.size(); // Returns a size of 2
```

Lastly, we can also remove values from a vector by using the `pop_back()` function. **Now there is a limitation of the `pop_back` function...which is that we can only delete the LAST value from a vector** (a.k.a from the back of the vector). Therefore, if we use the following, we will only be able to remove the value 2:

```cpp
myVector.pop_back(); // Removes the value 2 from the vector
```

Besides that, most of the vector functionality is the same as that of an array. **We can still access elements that already exist by position.** This can be used for access or modification of the position in the vector. For example, if we wanted to change the first position of the current array to the value 100, we can still do `myVector[0] = 100;`

---

**OTHER IMPORTANT TOPICS**

*String and CString Functions*

One common thing that I have been asked in almost every one of my sessions mentions something about string access or manipulation. **Within the string library there are so many different functions that are available to achieve these types of access.** All of these functions are located within the
**Standard Library for Strings**: Listed below are the most important functions to know about for this class.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>toupper(c)</td>
<td>If c is a lower case alphabetic character (a-z), returns the uppercase version (A-Z). If c is not a lower case alphabetic character, just returns c.</td>
</tr>
<tr>
<td>tolower(c)</td>
<td>If c is an upper case alphabetic character (A-Z), returns the lower case version (a-z). If c is not an upper case alphabetic character, just returns c.</td>
</tr>
<tr>
<td>strcpy()</td>
<td>Copies sourceStr (up to and including null character) to destStr.</td>
</tr>
<tr>
<td>strncpy()</td>
<td>Copies up to numChars characters.</td>
</tr>
<tr>
<td>strcat()</td>
<td>Copies sourceStr (up to and including null character) to end of destStr (starting at destStr's null character).</td>
</tr>
<tr>
<td>strncat()</td>
<td>Copies up to numChars characters to destStr's end, then appends null character.</td>
</tr>
<tr>
<td>strchr()</td>
<td>Returns NULL if searchChar does not exist in sourceStr. (Else, returns address of first occurrence, discussed elsewhere). NULL is defined in the string library.</td>
</tr>
<tr>
<td>strlen()</td>
<td>Returns number of characters in sourceStr up to, but not including, first null character. size_t is integer type.</td>
</tr>
<tr>
<td>strcmp()</td>
<td>Returns 0 if str1 and str2 are equal, non-zero if they differ.</td>
</tr>
</tbody>
</table>

```c
strcpy(destStr, sourceStr);  // Copies sourceStr (up to and including null character) to destStr.
strncpy(destStr, sourceStr, numChars);  // Copies up to numChars characters.
strcat(destStr, sourceStr);  // Copies sourceStr (up to and including null character) to end of destStr (starting at destStr's null character).
strncat(destStr, sourceStr, numChars);  // Copies up to numChars characters to destStr's end, then appends null character.
strchr(sourceStr, searchChar);  // Returns NULL if searchChar does not exist in sourceStr. (Else, returns address of first occurrence, discussed elsewhere). NULL is defined in the string library.
```
### CHECK YOUR LEARNING

1. If we wanted to know if the 3rd character within the string, `myString`, is an uppercase character, how would we check that?
2. What is a limitation of the `pop_back()` function?
3. True or False: The positions of a vector can be accessed the exact same as those of arrays.
THINGS STUDENTS MAY STRUGGLE WITH

1. **Note that vectors start at an index of 0, just like arrays.** This is super important to remember due to making sure you don’t cause issues within your program. For most looping and such, the size() function will work perfectly to make sure we are looping over the correct number of elements, since it changes with how many elements are in the vector automatically. However, understanding this is important for accessing the elements of the vectors correctly.

2. **Vectors are able to support all the same data types that arrays can.** This includes, ints, doubles, strings, chars, etc...

3. **Take time to get used to the different string manipulation and mutation functions...** Sometimes they won’t behave like you would like them to and need a little bit of playing around with to fully understand. Understanding string access and manipulation is a common coding area in computer science and software engineering interviews, so there is some merit to diving into understanding their behavior.

Answers:

1. isUpper(myString[2]); //Remember that the 3rd element, is the 2nd position
2. pop_back() can only delete the last element of a vector
3. True

Note: All tables were taken from the Baylor CS 1430 zyBook