CS 121 – Week 6 Worksheet – Functions Review – Solutions

1. What are the parts of a function definition? Give an example and identify those parts.

   Function to focus on:

```
int thisIsMyCoolFunction(double some_param, string another_param = “hello”)
{
  cout << “I have no idea what I am doing: “ << some_param << “ “ << another_param;
  return -1;
}
```

   Function header:

```
int thisIsMyCoolFunction(double some_param, string another_param = “hello”)
```

   Function return type (first thing in the function definition): `int`

   Function identifier/name: `thisIsMyCoolFunction`

   Parameter list (parenthesis portion of the function header):

```
(double some_param, string another_param = “hello”)
```

   Function body (and, if need to specify, the return statement is “return -1”):

```
{
  cout << “I have no idea what I am doing: “ << some_param << “ “ << another_param;
  return -1;
}
```

2. Give a definition example of a void function that prints a string and, if the function caller does not supply a string, prints "NO_MESSAGE_SUPPLIED".

```
void printString(string str = “NO_MESSAGE_SUPPLIED”)
{
  cout << str << endl;
}
```

3. Explain how the return keyword works.

   The return keyword works by terminating the execution of a function. While returning, the function will send either a variable or literal value that corresponds to the type of function it is (e.g. “return 0” or “return (a == b)”).

4. Can a void function have a return statement inside of it? If not, why is that? If so, how does it work?

   A void function can have a return statement, however it does not return anything and simply exits the function.

   Example:

```
void myFunct(bool printHello)
{
  if(!printHello)
  {
    return;  // exits function
  }
  else
  {
    cout << “Hello\n”;
  }
}
```

<SOLUTIONS CONTINUE ON NEXT PAGE>

5. Observe the following program and its output, and make guesses on why that's occurring. Afterwards, try to think of a way to prevent the array from being edited after calling `printArrayMultByTwo`.

PROGRAM	OUTPUT

`#include <iostream>`
`using namespace std;`
 	
`void printArrayMultByTwo(int int_list[], const int SIZE)`
`{`
`  for(int i = 0; i < SIZE; ++i)`
`  {`
`  int_list[i] *= 2;`
`  cout << int_list[i] << " ";`
`  }`
`  cout << endl << endl;`
`}`
 	
`int main()`
`{`
`  const int LIST_SIZE = 5;`
`  int my_list[] = {27, 32, 55, 1, 3752};`
`  printArrayMultByTwo(my_list, LIST_SIZE);`
`  printArrayMultByTwo(my_list, LIST_SIZE);`
`  printArrayMultByTwo(my_list, LIST_SIZE);`
 	
`  return 0;`
`}`
	`54 64 110 2 7504`
	
	`108 128 220 4 15008`
	
	`216 256 440 8 30016`
	
	Arrays are pointers and are passed by pointer (which is similar to pass by reference). This topic will be discussed in further detail when we reach pointers.
	
	A way to fix this is by setting a temporary variable to “int_list[i] * 2”. Example:
	
	`int x;`
	`for (int i = 0; i < SIZE; ++i)`
	`{`
	`  x = int_list[i] * 2;`
	`  cout << x << endl;`
	`}`
	
	Alternatively, you can just print the product:
	
	`for (int i = 0; i < SIZE; ++i)`
	`{`
	`  cout << (int_list[i] * 2) << endl;`
	`}`
	
	For added protection to the array, you can also change printArrayMultByTwo's parameters to:
	
	`(const int_list[], const int SIZE)`

6. (From the book) You know that the `==` operator can be used to test if two string objects are equal. However, recall that the comparison is case sensitive (e.g. "Billy" and "billy" are not equal strings).

   To remedy this, **write a complete program** that asks a user to enter two names and stores them in string objects. It should then report whether or not, ignoring case, they are the same.

   Some notes before you begin:
   - No input validation is required and you can request the names in any way you would like, however it is required for you to use `getline` to grab the names.
   - Remember to include any libraries that may be required.
   - To help the program accomplish its task, it should use two functions in addition to main: upperCaseIt() and sameString(). Here are the function prototypes:

```
string upperCaseIt(string s);
bool sameString(string s1, string s2);
```

An example solution of this problem is included on the GitHub repository (under the Week 6 Solutions Folder). It is labeled “str-eq-wk6.cpp”