CS 121 – Week 4 pt 2 Worksheet – Static Variables, Modulus, & Post/Pre-Increment/Decrement

1. Using the below program and output, explain how *static* works.

PROGRAM	OUTPUT

`#include <iostream>`
`using namespace std;`
`void mySpecialFunct()`
`{`
`  static int calls = 0;`
`  calls++;`
`  cout << "Function calls: "<< calls << endl;`
`}`
`int main()`
`{`
`  mySpecialFunct();`
`  mySpecialFunct();`
`  mySpecialFunct();`
`  mySpecialFunct();`
`  mySpecialFunct();`
`  mySpecialFunct();`
 	
`  return 0;`
`}`
	Function calls: 1
	Function calls: 2
	Function calls: 3
	Function calls: 4
	Function calls: 5
	Function calls: 6

2. Suppose you're testing running times of a computationally expensive function. After one run of your program, the total elapsed time for the run was 4100 seconds (ouch).

   If we wanted to see how long it took in terms of hours, minutes, and seconds, we would do the following:

   hours = 4100 secs / 3600 secs/hour = 1 hour R 500 secs

   minutes = 500 secs / 60 secs/minute = 8 minutes R 20 secs

   seconds = 20 seconds

   Which yields:

   1 hour, 8 minutes, and 20 seconds.

   Although the above is fairly straightforward arithmetic, you'd rather focus on optimizing your main program. To avoid being bothered with recalculations, **write a function that takes in an integer amount of seconds and outputs it in terms of hours, minutes, and seconds.** Output the conversion within your function in a way that you would prefer most.

3. What is the output of the following program?

PROGRAM	OUTPUT

`#include <iostream>`
`using namespace std;`
`int main()`
`{`
`  int counter = 5, sum = 0, c = 0.5;`
`  while(--counter > 0)`
`  {`
`  cout << ++sum << endl;`
`  }`
 	
`  counter = 5;  // reset counter`
 	
`  while(counter-- > 0)`
`  {`
`  cout << c++ << endl; // pun intended`
`  }`
`  return 0;`
`}`	

4. Analyze the following code and output. What's going in in it? **HINT:** All "%3" parts relate to the size of the list array.

PROGRAM	OUTPUT

`#include <iostream>`
`using namespace std;`
`int main()`
`{`
`  int list[] = {0,1,1};`
`  int n = 7, x;`
 	
`  cout << "INITIAL: \n[" << list[0] << " " << list[1]`
`  << " " << list[2] << "]\n\n";`
 	
`  cout << "LOOP:\n";`
`  for(x = 3; x <= n; ++x)`
`  {`
`  static int y = x-1, z = x-2;`
 	
`  list[x%3] = list[y%3] + list[z%3];`
`  y++;`
`  z++;`
 	
`  cout << "[" << list[0] << " " << list[1]`
`  << " " << list[2] << "]\n";`
`  }`
 	
`  x--; // doing this since the for loop shifted x up one`
 	
`  cout << "Value when n=" << n << ": " << list[x%3] << "\n";`
 	
`  return 0;`
`}`
	INITIAL:
	[0 1 1]
	
	LOOP:
	[2 1 1]
	[2 3 1]
	[2 3 5]
	[8 3 5]
	[8 13 5]
	Value when n=7: 13