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

```
#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;
}
```

OUTPUT

```
Function calls: 1
Function calls: 2
Function calls: 3
Function calls: 4
Function calls: 5
Function calls: 6
```

Static variables work in the following ways:
- It exists for as long as the program is on
- It only exists in the function it is defined
- A static variable does not lose its value when the program exits from the function
- They are a great alternative to global variables (as they can only be edited by one part of the program (the function it is in)

2. **Write a function that takes in an integer amount of seconds and outputs it in terms of hours, minutes, and seconds.**

Example function and output:

Program

```
#include <iostream>
using std::cout;

void outputSecToHMS (int seconds)
{
  int hours, minutes, remainder; // remainder used for "leftover" parts
  hours = seconds / 3600;        // 3600 seconds per hour
  remainder = seconds % 3600;    // get leftover seconds
  minutes = remainder / 60;      // 60 seconds per minute
  seconds = remainder % 60;      // the leftover now is just seconds

  cout << hours << " hour(s), " << minutes << " minute(s), "
       << seconds << " second(s)\n";
}
int main()
{
  outputSecToHMS(4100);
  outputSecToHMS(60);
  outputSecToHMS(3661);
  return 0;
}
```

Output

```
1 hour(s), 8 minute(s), 20 second(s)
0 hour(s), 1 minute(s), 0 second(s)
1 hour(s), 1 minute(s), 1 second(s)
```

3. What is the output of the following program?

PROGRAM

```
#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;
}
```

OUTPUT

```
1
2
3
4
0
1
2
3
4

NOTE:

pre-inc/dec acts before print and > check

post-inc/dec acts after print and > check
```

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

```
#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;
}
```

OUTPUT

```
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
```

EXPLANATIONS:
- This program is computing the nth term of the fibonacci question
- The program is “cycling” through the array
- The best way to understand how this program works is to write out every iteration of the for loop.
- Note that the static part does not reinitialize/redeclare y and z (only done once)