CPSC 331 — Supplement for Practice Midterm Test

The ***Slytherin Series*** is the sequence of integers $S_0, S_1, S_2, \ldots$ such that, for $n \geq 0$,

$$S_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ 2 \times S_{n-1} - S_{n-2} & \text{if } n \geq 2. \end{cases}$$

Algorithms for the following computational problem will be considered on this test.

Slytherin Number Computation

Precondition: A nonnegative integer n is given as input.

Postcondition: The n^{th} Slytherin number, S_n, is returned as output.

The following ***recursive*** algorithm for this problem is considered in the first two questions on this test.

```
int slytherin (n: int) {
1.  if (n==0) {
2.    return 0
3.  } else if (n==1) {
4.    return 1
    } else {
5.    return 2 × slytherin(n − 1) − slytherin(n − 2)
    }
}
```

The next algorithm, with a `while` loop is considered in the remaining questions on this test.

```
int cSlytherin (n: int) {
 1. if (n==0) {
 2.   return 0
 3. } else if (n==1) {
 4.   return 1
    } else {
 5.   int hocus = 0;
 6.   int pocus = 1;
 7.   int i = 1;
 8.   while (i < n) {
 9.     int shazam = hocus;
10.     hocus = pocus;
11.     pocus = 2 × hocus − shazam;
12.     i = i + 1;
      };
13.   return pocus;
    }
}
```