A triangular number is one whose value can be represented by an equilateral triangular arrangement of dots. The first few triangular numbers are:

1:  *      3:  *     6:     *      10:     *
    * *          * *            * *           * *
    * * *          * * *            *
    * * * *

The following incomplete C function is intended to test a value to see if it's a triangular number:

```c
__________ isTriangular(uint16_t Value) {   // 1
    uint16_t Sum = ___;                     // 2
    uint16_t Step = 1;                       // 3
    while ( Sum <= ____________ ) {          // 4
        Sum = Sum + Step;                     // 5
        if ( Value == Sum )                   // 6
            return true;                       // 7
        ++Step;                               // 8
    }
    return _________;                        // 9
}
```

For questions 1 through 4, consider completing the given function by filling the blanks.

1. How should the blank in line 1 be filled?
   1) void 3) boolean 5) None of these
   2) bool 4) uint16_t

2. How should the blank in line 2 be filled?
   1) Sum does not need to be initialized. 3) 1 5) None of these
   2) 0 4) Value

3. How should the blank in line 4 be filled?
   1) 0 3) Value 5) None of these
   2) 1 4) 0x0000FFFF

4. How should the blank in line 9 be filled?
   1) 0 3) true 5) None of these
   2) 1 4) false
The following incomplete C function is intended to swap the values of its two int parameters. That is, if the caller has two int variables A and B, a call Swap() can be used to swap the values of A and B.

```c
void Swap(int x, int y) {  // 1
    int temp = \__________; // 2  Back up first integer
    \__________;           // 3
    \__________ = temp;    // 4
}
```

For questions 5 through 7, consider completing the given function by filling the blanks.

5. How should the blanks in line 1 be filled? (Both should be filled the same way.)

   1) void  
   2) int  
   3) int*  
   4) 2 or 3 only  
   5) None of these

6. How should the blank in line 2 be filled?

   1) x  
   2) y  
   3) *x  
   4) *y  
   5) None of these

7. How should the blank in line 3 be filled?

   1) x = y  
   2) y = x  
   3) *x = *y  
   4) *y = *x  
   5) None of these

8. Suppose you have an implementation of the following function:

    ```c
    // Pre: N has been initialized
    // Post: the value of N! has been returned
    uint64_t factorial(uint32_t N);
    ```

    Assuming you have the following variables in scope:

    ```c
    uint32_t N = 10;
    uint64_t facN;
    ```

    which of the following is the best way to call the function factorial() to find the value of 10!?

   1) factorial(10);  
   2) factorial(N);  
   3) facN = factorial(10);  
   4) facN = factorial(N);  
   5) All of them would be fine.  
   6) 1 and 2 only  
   7) 3 and 4 only  
   8) 1 and 3 only  
   9) 2 and 4 only  
   10) None; none of the given choices will work.
9. The following function calculates the N-th Fibonacci number:

```c
//     N:  0   1   2   3   4   5   ...  
// FiboN:  1   1   2   3   5   8   ... 
void fibonacci(uint64_t* FiboN, uint32_t N) {
    if ( N <= 1 ) *FiboN = 1;
    uint64_t fNMinusTwo = 0;
    uint64_t fNMinusOne = 1;
    uint64_t fN;
    for (uint32_t i = 2; i <= N; i++) {
        fN = fNMinusOne + fNMinusTwo;
        fNMinusTwo = fNMinusOne;
        fNMinusOne = fN;
    }
    *FiboN = fN;
}
```

Suppose the caller has the following variables in scope:

```c
guint32_t N = 8;
guint64_t FiboN;
```

Which of the following would be the best way to make a call to the function `fibonacci()`?

1) `fibonacci(FiboN, N)`
2) `fibonacci(&FiboN, N)`
3) `fibonacci(&FiboN, &N)`
4) `fibonacci(*FiboN, N)`
5) `fibonacci(*FiboN, *N)`
6) None of these

10. If a C function has a return type that is not `void`, then:

1) The function should not take any parameters.
2) The function must take parameters.
3) Calls to the function should always be C statements, like: \( F(...) \);
4) Calls to the function should be the right side of an assignment statement, or occur as part of an expression.
5) None of these