Question 1. True or False

Circle **T** if the statement is true, otherwise circle **F** if the statement is false.

1. Generic programming is a subset of metaprogramming **T** **F**

2. If template specialization is not used (i.e. not instantiated), its code is not generated for the final executable. **T** **F**

3. For template `T foo()`, you can write `int a = foo()` to instantiate the function template `foo` with an `int` parameter . **T** **F**

4. Default template arguments must be compile-time constants. **T** **F**

5. An iterator always requires an associated iterable. **T** **F**

Question 2. Short Answers

a) List 3 differences between Rust generics and C++ templates.

b) Explain the difference between iter(), iter_mut(), and into_iter().

Question 3. Programming Questions

1. In Rust, create a generic struct named Factorial and implement the iterator trait for it.

2. Implement the binary search algorithm using a C++ function template, assume the array is sorted and return -1 upon not found.

```
template<typename T>  /* find index of val in array of size n */
int binary_search(const T & val, T * array, int n) {









}
```

3. Implement a C++ template class named Triple that is a tuple of 3 elements of the same type. Overload enough operators so that the binary search template you implemented above can be instantiated for Triple. Use lexicographical order.

4. Write a C++ template class, `Pair`, which accepts one template parameter, `T`, and creates a class which has two fields, `x` and `y`, both of type `T`. Implement a constructor that takes two parameters to initialize both `x` and `y`, and implement a member function named `get` which returns a pointer to `x` if the single integer argument is 0, pointer to `y` if the argument is 1, and `nullptr` for all other argument values. You are required to implement the member function `get` outside of the body of the template class (i.e. you may not inline the member function inside the class).

5. Create a generic Queue class using templates. Implement the Queue using a singly linked list, with the member functions, `push_back`, that pushes new elements to end of the queue, `front`, which returns the first element of the queue, and `pop_front`, which removes the first element of the queue.

6. Write a recursive template named SumSquare that will calculate the sum of squares of consecutive integers from 1 to N. e.g. `SumSquare<5>::value` should be 55.