**Code Test: *Heat Map Service***

Copyright © 2010 ArenaNet

All Rights Reserved. Confidential Property of ArenaNet, LLC

**Objective**: Design and implement a heat map library that collects counter-based data and tracks regional “hot spots” over time in a two-dimensional map.

**Description**: It is often helpful in a map-based game like an MMO to collect statistical data based on its location in the world. For instance, designers find it useful to know where the most popular map locations are so that they can adjust spawns and encourage players to visit other parts of the map. We would like you to design a flexible heat map library to collect and aggregate counter-based data and organize it spatially.

**Requirements of the library:**

*Unknown map size at instantiation.* At the time the heat map is instantiated, you will not necessarily know its full dimensions.

*Known resolution at instantiation*. You may assume the heat map’s spatial resolution (i.e., the smallest spatial unit of data collection) is known at instantiation time. The resolution will be passed as a parameter.

*Multiple counters*. The library must be capable of tracking multiple counters (e.g., the number of players who died at a location, the amount of gold collected at a location, etc.).

*Simple accumulator-based counters*. Each heat map location counter requires only a simple accumulator. You do not need to track counters over multiple time frames.

*Serialization*. Your library must be capable of serializing and deserializing heat map data to and from a buffer. Serialization should make it possible to persist and restore the state of a heat map instance. The user of the library may choose to store and load the serialized data however he or she pleases. Apart from serialization, you do not need to worry about networking or data storage.

*Query interface*. The user of the library should be able to query an entire grid’s worth of data for a given counter.

**Expectations for your implementation:**

*Efficient.* Your library should be efficient in both time and space.

*Robust*. Your library should be bug-free.

*Correct.* Your library should produce accurate results.

*Well-designed.* Your library, particularly its API, should be easily understood by any programmer who uses it.

**What to submit**: Implement your API and library in C++. It should be designed to be included as a static library in both the game and tools to analyze the game. You may NOT use STL. Please submit a complete project that can be compiled by Microsoft Visual Studio (preferred) or gcc. If you write any tests, please include them.

**What to do if you have questions**: If there is any aspect of this problem that you feel is not fully specified, please feel free to provide your own additional specifications.