CS7491. Fall 08. Midterm. NAME: _____ Grade:_____ / 20.

Individual test. Sit sufficiently far from other students and do not look at other students' work. Closed books. Please type and write legibly. Each question is 10 points. Allocate your time judiciously.

1) **Visibility**: You have access to an approximation of a 3D terrain (height-field), represented by a triangle mesh stored in Corner Table format (G, V, O arrays). The z-direction is the height. As input, you are given the (x,y) locations of two points, P_0 and P_1, on the terrain. You are also given the ids, t_0 and t_1, of the triangles whose projection on the X-Y plane contains the projection of the two corresponding point. Answer the following questions by using the corner table operators (triangle t(c), next clockwise corner n(c) in t(c), previous p(c), opposite o(c), id v(c) of vertex, geometric location g(c) of vertex, b(c) true when c has no opposite) and simple geometric constructions (points, vectors, •, ×…). Your solutions must be efficient, in that they should only access the necessary triangles, corners, and vertices, and **never traverse the entire mesh**.

(1-a) Explain (provide geometric construction or detailed code) for infering the height (z-coordinate) $P_0.z$ of P_0 from the height of the vertices of t_0.

(1-b) Assume that one walks from P_0 to P_1 along a path that projects onto a straight line-segment in the X-Y plane. Provide a detailed algorithm (in Processing or pseudocode) for accessing, one by one, the triangles be visited by this walk. Discuss the starting and ending conditions, and the details of an efficient geometric tests to advance to the next triangle.

(1-c) Explain how the above solutions can be used to test whether P_1 is **visible** (seen by looking above ground) from P_0. You do not need to provide a detailed pseudocode, but must ensure that the reader clearly sees that you know how to do this.

(1-d) What would you do if the ids of t_0 and t_1 were not given? Suggest an efficient way of solving the above visibility problem without a priori knowledge of the ids of t_0 and t_1 that, in most cases, only accesses a small fraction of the triangles.

2) **Compression**: Consider the following algorithm for compressing the connectivity of any triangle mesh that is a single, zero-genus, water-tight (no border) manifold shell. We initialize ***Land*** to be triangle T_0. Then, we grow Land by attaching one at a time a new triangle to its border edges, but only when this triangle has a tip vertex not on the border of the Land (similar to the C case in EdgeBreaker). We stop when no more such triangles can be attached. The remaining triangles form the ***Water***. The ***Beach*** is the loop of edges separating Water from Land. (The figure, below illustrates the concept of Land (left) and Water (right) for a small mesh **with border**. Note that these do not accurately represent the problem, since your mesh will have no border.)

(2-a) Prove that the Beach forms a **single manifold loop** and **explain** why this **implies** that Land and Water are each **simply connected** (which is not the case in the figure below, drawn for a mesh with border).

We encode vertices in the order in which they appear along the Beach (starting from an arbitrarily chosen place along the cyclic loop). Then, we pick the first edge of the Beach as ***gate*** (red) and use EdgeBreaker to encode the connectivity of the Land (pretending that Water has already been encoded and that the Beach is bounding the yet-to-be-encoded region and that all its vertices have been visited).

(2.c) Which EdgeBreaker symbol(s) will **never** be **used**? **Justify** your answer.

(2.d) Suggest a trivial binary **coding** for the EdgeBreaker symbols used when compressing the connectivity of the Land that guarantees no more than **2 bits per triangle** of the **Land**.

The proposed algorithm simply points out that the same technique may be used to encode the connectivity of the Water (starting from the same gate, shown in red below), hence guaranteeing less than 2 bits per triangle. Since each triangle knows which border vertices it spans, we have the V table.

(2.e) Is this algorithm **correct**? If so, **prove** it. If not, clearly explain (and draw) a simple situation when it will **fail**.

Assume that the above algorithm has worked correctly on a given mesh of T triangles (as shown below-right).

(2.f) What can you say about the **number of triangles** in the Land and in the Water? **Prove** your answer.