**Project**

**Hash Tables and Binary Files**

In this project, you may use BinaryFileResources.zip under Resources as a project resource. The data for your table will use a WhoopingCrane class with data members of ID (a unique whole number), location (text), age (double), and gender (male or female). You will created this class.

In working with your hash table, you will use a binary file (previously we have used text files in our work). A good resource for understanding binary files that save object data with text data is the following video: <https://www.youtube.com/watch?v=P7XGOBoVzW4> (also see CSC 231 Units for additional resources).

This project will include a menu which has options (1) for adding a new whooping crane to the hash table, (2) searching for a specific whooping crane by ID number, (3) display all whooping cranes in a user-specified area, (4) removing a whooping crane from the hash table, and (5) displaying the hash table with each whooping crane’s position in the table.

Your program should also: (6) add six code-created whooping cranes to your hash table as the program begins and use a hash table with a size of 19, and (7) your hash table should use quadratic probing to resolve collisions.

Additionally, create a test report in a Word document for your program that (8) shows an initial display of your hash table (with the six code-created whooping cranes) and then four sequential screenshots showing the results of adding two whooping cranes and removing two whooping cranes from the table. Annotate your screenshots for clarity. Also (9) in an additional annotated screenshot explain the quadratic probing process for how a specific whooping crane is added to your hash table with at least two collisions. (10) Include a screenshot also for each of the following: your add process, the process and results of your search for a specific whooping crane, and the process and results of your search for whooping cranes in a specific location.

Submission instructions:

1. Submit your .zip zipped project folder (do not use .rar files or zip files within zip files and do not simply turn in .cpp and .h files).
2. Turn in a project that compiles and runs. Comment out parts of your program that do not compile or run.
3. Submit Word document as a separate attachment, separate from your project folder. This Word document will contain all screenshots and explain which assigned program features work and which do not. If any do not work, show code that doesn’t work for partial credit with a brief explanation.

Grading: Your program should be well-organized with multiple files, have clean displays and menus, and use meaningful identifiers (if not appropriate points will be taken off). More generally, items 1 through 10 specified above are worth 10 points each for a total of 100 points.

Helpful code segment:

