Sabrina Oh

CS302. Data Structures

Project Design Document

* Node/City class
  + Members
    - Constructor (default, parameterized, copy)
    - Destructor
  + Attributes
    - Name of City (string)
    - Distance to adjacent cities (int)
    - Pointers to adjacent cities (City Class\*)

|  |  |
| --- | --- |
| UML Notation | Functionality |
| getName() : string | Returns the name of the city |
| setName (newName) : void | Adds or changes the name of a city |
| getDistance() : int | Returns the distance between the city and one of its pointers |
| setDistance(newDistance : int) : void | Sets the distance from the city to its adjacent |
| addEdge(endName : string, int weight) : void | Adds an edge to another city |
| getWeight (endName : string) : void | Returns the weight of an edge |

* Stack Class/Functionality
  + Attributes
    - Pointer for the top of the stack (CityClass\*)

|  |  |
| --- | --- |
| UML Notation | Functionality |
| peek() : NodeType | Gets copy of the top object in the stack |
| push(newData: NodeType): void | Adds an object to the top of the stack |
| pop() : NodeType | Removes and returns a copy of the top object of the stack |
| clear() : void | Clears the objects in the stack |
| isEmpty() : bool | Checks if the stack has any objects |
| isFull() : bool | Checks if the stack has the maximum number of objects |

* Graph Class
  + Attributes

|  |  |
| --- | --- |
| UML Notation | Functionality |
| isEmpty(CityClass city) : bool | Returns true if graph empty, false if not |
| getNumVertices(CityClass city) : int | Returns the number of vertices in a graph |
| getNumEdges(CityClass city) : int | Returns the number of edges to a city |
| add(ItemType start, ItemType end, int edgeweight): void | If the city name does not exist create a vertex |
| Remove(ItemType start, ItemType end) : bool | Returns true if the item is removed, otherwise false |
| getEdgeWeight(ItemType start, ItemType end): int | Returns the weight of the edge (the length) |
| depthFirstTraversal (ItemType start, void visit(ItemType&)) | Should have a public and a protected method. The function listed here is the protected method.  The private method will only take in a start. This traverses the graph starting at the start specified, in a depth first manner. It will output the output the order of cities visited |
| breadthFirstSearch(ItemType start, void visit(ItemType&)) | Should have a public and a private function. The function listed here is the protected method. This function takes in a start location, and traverses the graph in a breadth first manner. It will output the order of cities visited |

* Graph Interface Class
  + This interface is templated, but all the functions it contains are pure virtual functions
  + Inherits from Graph Class

|  |  |
| --- | --- |
| UML Notation | Functionality |
| virtual int getNumVertices() const =0 |  |
| virtual int getNumEdges() const = 0 |  |
| virtual bool add (ItemType start, ItemType end, int edgeWeight) = 0 |  |
| virtual bool remove (ItemType start, ItemType end) const = 0 |  |
| virtual int getEdgeWeight (ItemType start, ItemType end) const = 0 |  |
| virtual void depthFirstTraversal (ItemType start, void visit (ItemType&)) |  |
| virtual void breadthFirstTraversal (ItemType start, void visit (ItemType&)) |  |
| virtual ~GraphInterface() {} |  |

* Adjacency Matrix
  + Numbers in parenthesis is approximate driving distance (edge weight)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Reno | San Francisco | Salt Lake City | Seattle | Las Vegas |
| Reno | - | 1 (218) | 1 (518) | 1 (704) | 1 (443) |
| San Francisco | 1 (218) | - | 0 | 1 (808) | 1 (564) |
| Salt Lake City | 1 (518) | 0 | - | 1 (840) | 1 (425) |
| Seattle | 1 (704) | 1 (808) | 1 (840) | - | 0 |
| Las Vegas | 1 (443) | 1 (564) | 1 (425) | 0 | - |

Main Driver

* Functionality
* Outside of the main function
  + A templated visit function, return type void
    - Takes in a templated item, and prints to the screen the item

Responsibilities

Timeline

Communication

* Will communicate primarily through Discord
* Phone contact information has been exchanged in the event we are not able to contact through Discord

Technologies Used