ECE 250 - PROJECT 2 - DESIGN DOCUMENT

ARJUN MEHTA – [redacted]

OVERVIEW OF CLASSES

class cityinfo

Represents the class that stores the name, latitude, longitude, population, cost of living and the average net salary of the city.

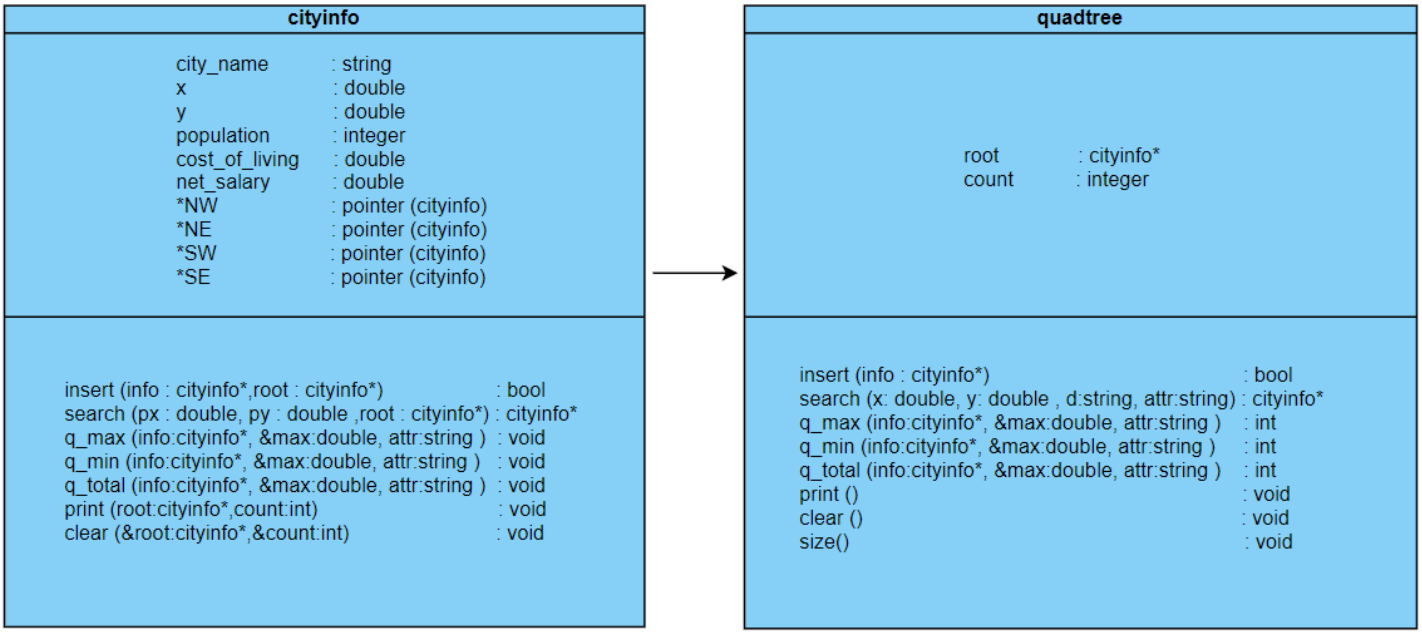
Member Variables :	city_name (string)	**stores the name of the city**
	x (double)	**stores the longitude of the city**
	y (double)	**stores the latitude of the city**
	population (int)	**stores the population of the city**
	cost_of_living (double)	**stores the cost of living in the city**
	net_salary (double)	**stores the average net salary in the city**
	*NE (pointer)	**pointer of type cityinfo that represents the north-east direction**
	*NW (pointer)	**pointer of type cityinfo that represents the north-west direction**
	*SW (pointer)	**pointer of type cityinfo that represents the south-west direction**
	*SE (pointer)	**pointer of type cityinfo that represents the south-east direction**
Member Functions :	insert()	**inserts a new city, called by quadtree::insert (recursive approach)**
	search()	**searches for a new city, called by quadtree::search (recursive approach)**
	q_max()	**finds maximum value of attribute in direction d, called by quadtree::q_max**
	q_min()	**finds minimum value of attribute in direction d, called by quadtree::q_min**
	q_total()	**finds total value of attribute in direction d, called by quadtree::q_total**
	print()	**prints the entire quadtree, called by quadtree::print (recursive approach)**
	clear()	**clears the entire quadtree, called by quadtree::clear (recursive approach)**

class quadtree

Represents the class that forms the basis of the quadtree, defines the required data variables and member functions

Member Variables :	count (int)	**keeps track of the number of cities in the quadtree**
	*root (pointer)	**pointer of type cityinfo that represents the root of the quadtree**
Member Functions :	insert()	**inserts a new city, calls cityinfo::insert (recursive approach)**
	search()	**searches for a new city, calls cityinfo::search (recursive approach)**
	q_max()	**finds maximum value of attribute in direction d, calls cityinfo::q_max**
	q_min()	**finds minimum value of attribute in direction d, calls cityinfo::q_min**
	q_total()	**finds total value of attribute in direction d, calls cityinfo::q_total**
	print()	**prints the entire quadtree, calls cityinfo::print (recursive approach)**
	clear()	**clears the entire quadtree, calls cityinfo::clear (recursive approach)**
	size()	**displays the size of quadtree**

UML CLASS DIAGRAM



DETAILS ON DESIGN DECISIONS

Constructors

class cityinfo
Set default values for name, latitude, longitude, population, cost of living and the average net salary of the city.
Set NE, NW, SW and SE = nullptr so that a newly created node always points towards a null pointer.

class quadtree
Set count equal to 0.
Set root = nullptr so that a newly created node always points towards a null pointer.

Destructors

class quadtree
Run the clear() function inside destructor to completely clear the quadtree to avoid memory leaks.

TEST CASES

We implement various test cases to check if program is able to handle difficult situations. This includes running each and every function, exiting before the final statements and check for failures in cases where overlapping takes place.

size i City;43.7;54.9;12000;1290;2700 i GhostTown;43;56;12989;900;1500 i Pembroke;34;65;25000;1000,1200 s 43.7;54.9 size print q_max 43.7;54.9,NE;r clear print size exit	i Delhi;43.7;54.9;12000;1290;2700 i Mumbai;23;21;12989;900;1500 i Waterloo;34;65;25000;1000,1200 size print q_max 43.7;54.9,NE;r q_min 43.7;54.9,NW;p q_total 43.7;54.9,SE,s i Rio;43.7;54.9;12000;1290;2700 clear print i London;98;-23;12000;129;270 print size exit	i Toronto;43;54;19800;1290;2700 i Dubai;-99;-54;12000;1450;3100 i Mumbai;23;21;12989;900;1500 i Waterloo;34;65;25000;1000,1200 print size clear i Toronto;43;54;19800;1290;2700 i Dubai;-99;-54;12000;1450;3100 i Mumbai;23;21;12989;900;1500 i Waterloo;34;65;25000;1000,1200 s 34;65 q_total -99;-54,NW,p print exit size print
Checks size of empty quadtree, inserts cities, searches for city, checks size again and then prints. Finds max value of cost of living in direction NE from City. Clears and then prints empty quadtree. Checks if size is 0.	Inserts various cities, runs size and print. Checks output of q_max, q_min and q_total for different directions and attributes. Inserts city with same coordinates to check for failure.	Inserts cities, prints and checks size. This time we check if we're able to add negative coordinates (Dubai). Clears and tries to insert same cities again. Runs search and q_total. Exits before size and print to check if exit works properly.

PERFORMANCE CONSIDERATIONS

Time complexity and time taken by each function

Insert function : O(lg(n))
Clear function : O(n)

Achieved a time complexity of O(lg(n)) in the **insert** function by comparing the passed values of x and y to the x and y values of the root, choosing a direction and then recursively calling the insert method. We keep repeating this method till we find an empty position to put the given data of the city.

Achieved a time complexity of O(n) for **clear** function by traversing and recursively calling the clear method. We traverse till the end (with respect to one direction) delete the last node, move to the second direction and do the same. In the end we delete the root node and set it to nullptr to complete our clear function.

Note : We assume that the tree is balanced to justify the above mentioned explanation.