**Point2D – Function Definitions**

*Point2DImpl();*

The first constructor creates a Point2D*Impl* object but, only defines our pointer to our struct and doesn’t store any point information.

*Point2D(std::vector<RGPPoint2D> pointVector);*

The second constructor creates a Point2DImpl object based on a passed in vector of RGP points.

*Point2DImpl(std::string listOfPoint2DString);*

The third constructor creates a Point2DImpl object based on a passed in string of points.

*Point2DImpl(std::ifstream& file);*

The final constructor creates a Point2DImpl object based on a passed in file of points.

*~Point2DImpl();*

The destructor clears any data that is stored in our vector of points or our bounding box vector.

*std::string getPointString();*

Returns a string containing all of the points in our vector of points.

*void printAllPoints();*

Prints out every stored point in our vector of points.

*bool isEmptyPoint();*

Returns true if vector of points is empty, false otherwise.

*bool isValidPoint();*

Returns true if Point2D object is valid, false otherwise.

*int getNumberOfPoints();*

Returns the total number of points stored in our vector.

*Point2DImpl getBoundingBox();*

Returns Point2DImpl object with bounding box as a vector attribute.

*iterator begin();*

Returns an iterator defining the starting position of vector of points.

*iterator end();*

Returns an iterator defining the ending position of vector of points.

*iterator(RGPPoint2D\*);*

Assigns our RGPPoint2D pointer in iterator class to incoming parameter.

*RGPPoint2D operator\*();*

*RGPPoint2D operator++(int);*

*RGPPoint2D operator++();*

*bool operator!=(const iterator&);*

*bool operator==(const iterator&);*

*bool add(RGPPoint2D rgpp2d);*

Returns true if our passed in RGP point is successfully added to our vector of points.

*bool update(iterator it, RGPPoint2D rgpp2d);*

Returns true if our passed in RGP point is successfully added to our vector of points using an iterator as index, false otherwise.

*bool remove(iterator it);*

Returns true if successfully removed RGP point using iterator as index from vector of point, false otherwise.

*bool operator==(const Point2DImpl &p2d);*

Returns true if two Point2Ds are equal, false otherwise.

*bool operator!=(const Point2DImpl &p2d);*

Returns true if two Point2Ds are not equal, false otherwise.

*Point2DImpl operator[](int index);*

Returns a specified RGP point at specified index.

*Point2DImpl operator=(const Point2DImpl &p2dImpl);*

*void pointSort(std::vector<RGPPoint2D> &bar);*

Sorts a passed in vector of points into our stored vector of points.

*void mergeSort(std::vector<RGPPoint2D> &left, std::vector<RGPPoint2D> &right, std::vector<RGPPoint2D> &bars);*

Merges two passed in vector of points.

*bool parseStringToVectorOfPoints(std::string st);*

Returns true if the passed in string of points is successfully stored in vector of points, false otherwise.

**Line2D – Function Definitions**

*Line2DImpl();*

The first constructor creates a Line2DImpl object but, only defines our pointer to our struct and doesn’t store any segment information.

*Line2DImpl(std::vector<RGPHalfSegment2D> listOfSegments);*

The second constructor creates a Line2DImpl object based on a passed in vector of RGP half segments.

*Line2DImpl(std::string listOfLine2DString);*

The third constructor creates a Line2DImpl object based on a passed in string of segments.

*Line2DImpl(std::ifstream& file);*

The final constructor creates a Line2DImpl object based on a passed in file of segments.

*~Line2DImpl();*

The destructor clears any data that is stored in our vector of segments or our bounding box vector.

*std::string getLineString();*

Returns a string containing all of the line segments in our vector of segments.

*void printAllLines();*

Prints out every stored line in our vector of segments.

*bool isEmptyLine();*

Returns true if vector of segments is empty, false otherwise.

*bool isValidLine();*

Returns true if Line2D object is valid, false otherwise.

*int getNumberOfSegments();*

Returns the total number of segments stored in our vector.

*Line2DImpl getBoundingBox();*

Returns Line2DImpl object with bounding box as a vector attribute.

*iterator begin();*

Returns an iterator defining the starting position of vector of segments.

*iterator end();*

Returns an iterator defining the ending position of vector of segments.

*iterator(RGPHalfSegment2D\*);*

Assigns our RGPHalfSegment2D pointer in iterator class to incoming parameter.

*RGPHalfSegment2D operator\*();*

*RGPHalfSegment2D operator++(int);*

*RGPHalfSegment2D operator++();*

*bool operator!=(const iterator&);*

*bool operator==(const iterator&);*

*bool add(RGPHalfSegment2D rgpSeg2d);*

Returns true if our passed in RGP half segment is successfully added to our vector of segments.

*bool update(iterator it, RGPHalfSegment2D rgpSeg2d);*

Returns true if our passed in RGP half segment is successfully added to our vector of segments using an iterator as index, false otherwise.

*bool remove(iterator it);*

Returns true if successfully removed RGP half segment using iterator as index from vector of segments, false otherwise.

*bool operator==(const Line2DImpl &l2d);*

Returns true if two Line2Ds are equal, false otherwise.

*bool operator!=(const Line2DImpl &l2d);*

Returns true if two Line2Ds are not equal, false otherwise.

*Line2DImpl operator[](int index);*

Returns a specified RGP half segment at specified index.

*Line2DImpl operator=(const Line2DImpl &l2dImpl);*

*void lineSort(std::vector<RGPHalfSegment2D> &bar);*

Sorts a passed in vector of segments into our stored vector of segments.

*void mergeSort(std::vector<RGPHalfSegment2D> &left, std::vector<RGPHalfSegment2D> &right, std::vector<RGPHalfSegment2D> &bars);*

Merges two passed in vector of segments.

*bool parseStringToVectorOfLines(std::string st);*

Returns true if the passed in string of lines is successfully stored in vector of segments, false otherwise.

**Region2D – Function Definitions**

*Region2D();*

The first constructor creates a Region2D object but, only defines our pointer to our struct and doesn’t store any point information.

*Region2D(std::vector<std::vector<RGPAnnotatedHalfSegment2D>> listOfRegions);*

The second constructor creates a Region2D object based on a passed in vector of vectors of annotated half segments.

*Region2D(std::string listOfRegion2DString);*

The third constructor creates a Region2D object based on a passed in string of regions.

*Region2D(std::ifstream& file);*

The final constructor creates a Region2D object based on a passed in file of points.

*~Region2D();*

The destructor clears any data that is stored in our vector of vectors of annotated half segments or our bounding box vector.

*std::string getRegionString();*

Returns a string containing all of the regions in our vector of vectors of annotated half segments.

*Number area();*

Returns a Number object representing area of our region object.

*static bool isEmptyRegion(Region2D region);*

Returns true if our passed in region is empty, false otherwise.

*static bool isValidRegion(Region2D region);*

Returns true if our passed in region is a valid region, false otherwise.

*int getNumberOfFaces();*

Returns the number of faces for or region object.

*std::vector<RGPSegment2D> getBoundingBox();*

Returns the bounding box of our region object.

*bool operator==(const Region2D &p2d);*

Returns true if passed in region object is equal to region object, false otherwise.

*bool operator!=(const Region2D &p2d);*

Returns true if passed in region object is not equal to region object, false otherwise.

*std::vector<RGPSegment2D> getCycle(int index);*

Returns the cycles of a region at specified index.

*std::vector<RGPAnnotatedHalfSegment2D> getFace(int index);*

Returns a face of our region object.

*bool addFace(std::vector<RGPSegment2D>);*

Adds a face to a region object.

*bool update(int index, std::vector<RGPSegment2D>);*

Updates a region at a specified index

*bool remove(int index);*

Removes a region at a specified index.

*std::vector<RGPSegment2D> operator[](int index);*

Returns a region at a specified index.

*bool parseWDR(std::string inputString);*

*Number getAreaOfCycle(std::vector<RGPAnnotatedHalfSegment2D> vectorOfSegments);*

Returns the area of a passed in cycle.

*std::vector<RGPAnnotatedHalfSegment2D> constructRegion(std::string formattedInputString);*

Returns a constructed region based on passed in string.