CSE102 – Computer Programming with C (Spring 2020)

Summer Project – 2D CAD Library

Handed out: 9:00pm Wednesday July 22, 2020.

Due: Start of CSE241 in Fall 2020.

Hand-in Policy: Hand in directly to CSE241 course instructor (will let you know the submission details).
Collaboration Policy: No collaboration is permitted.
Grading: This project will contribute a certain amount to your coursework for CSE241 in Fall 2020. The exact contribution is to be decided by the course instructor.

Description: You will implement a C library to help create and manipulate 2D Computer Aided Design (CAD) content. Examples for 2D CAD software include https://www.qcad.org/ and https://librecad.org/.

Your library will have the following functions. Note that there may be more (conceptually overloaded) instances of these functions. A minimal subset is provided below. Some functionality is left open for you to define and implement. A question mark (?) in the function name implies that you are expected to define appropriate name(s) for multiple instances.

- **`CAD2D * c2d_start?():`**
  `CAD2D * c2d_start?(double width, double height):`
  `CAD2D * c2d_start?(double width, double height, const Hierarchy * h):`
  Initializes your CAD content. A CAD content is initialized optionally on a canvas of a given dimension (`width` × `height`). Anything that you draw on the canvas should be within this limit. Note that the coordinate system of your canvas starts from the bottom-left corner. x axis is towards left and y axis is perpendicular in the up direction. See the following figure for an illustration of the canvas geometry. If no canvas is initialized, assume no bounds. Every CAD entity belongs to a hierarchy. The default hierarchy is the highest level. One can change or reassign hierarchies. Hierarchies can be nested.

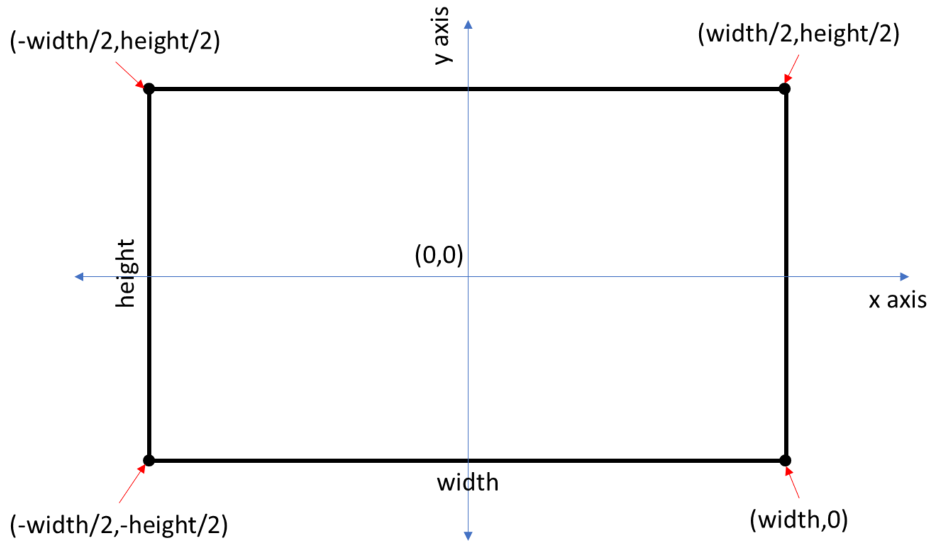

Figure 1 Canvas initialization with width and height and the assumed coordinate frame.

- **`Label * c2d_add_point?(CAD2D * cad, double x, double y):`**
  `Label * c2d_add_point?(CAD2D * cad, Point2D p):`
  `Label * c2d_add_point?(CAD2D * cad, Point2D p, const Hierarchy * h):`
  `Label * c2d_add_point?(CAD2D * cad, Point2D p, const Hierarchy * h, const Label * l):`

  Add a point to the current or the given context/hierarchy. This function returns a pointer to the label attached to the created point unless it is given by the calling function. See the note in the data structures related to the uniqueness of the labels.

- **`Label * c2d_add_<BASIC>?(CAD2D * cad, ...):`**
  `Label * c2d_add_<BASIC>?(CAD2D * cad, ...):`
  `Label * c2d_add_<BASIC>?(CAD2D * cad, ..., const Hierarchy * h):`
  `Label * c2d_add_<BASIC>?(CAD2D * cad, ..., const Hierarchy * h, const Label * l):`

  Similar to point adding, add a new basic entity. Basic entities can be lines, arcs, circles, ellipses, splines, polylines, texts, polygons (including filled versions) and images. You will need appropriate data structures (as arguments defining their properties) to set these basic entities. Assume that any 2D basic shape (including points) can have color, thickness and line style as well.

- **`void c2d_snap(CAD2D * cad, const Label * ls, const Label * lt):`**

  Snaps a given entity (in ls) to another entry (lt). The following rules apply:

LS	LT	Snap Action
Point	Point	Snaps the point in LS to point in LT.
Point	Line (or Polyline)	Snaps the point in LS to the closest end of the line(s).
Point	Polygon	Snaps the point to the center of the polygon.
Point	Arc	Snaps the point to the center of the arc.
Polyline	Point	Define a strategy!
Polyline	Polyline	Define a strategy!
Polyline	Polygon	Define a strategy!
Polygon	Point	Define a strategy!
Polygon	Polyline	Define a strategy!
Arc	Point	Define a strategy!
Arc	Polyline	Define a strategy!

- **`double c2d_measure(CAD2D * cad, const Label * ls, const Label * lt):`**

  Measure the distance between two entities as defined in the following table:

LS	LT	Snap Action
Point	Point	Euclidean distance between two points.
Point	Polyline	The distance closest line.

Point	Polygon	The distance to the closest edge of the polygon.
Point	Arc	The shortest distance to the arc.
Any Item	Any Item	The shortest distance between these 2D shapes.
Any Item	Any Item	The shortest distance between the centers of the 2D shapes. May need to define the center for each shape.

- **`Hierarchy * c2d_create_hierarchy?(CAD2D * cad):`**

  `Hierarchy * c2d_create_hierarchy?(CAD2D * cad, Hierarchy * parent):`

  Creates a hierarchy. A new hierarchy can be at the highest level or a child of a parent.
- **`void c2d_export(CAD2D * cad, char * file_name, char * options):`**

  Exports the current CAD to a file with the given options. The option indicates the targeted output file format. For now, consider:

  EPS: For the file format please refer to the attached document “Encapsulated PostScript File Format Specification Version 3.0.pdf”.

  GTU: Your own file format that can be imported.
- **`CAD2D * c2d_import(char * file_name, char * options):`**

  Import a CAD model from a given file. For now, consider only .GTU files for which you will define a format.

You are supposed to have at least the following data structures:

- **`CAD2D:`** The data structure to hold the current CAD content.
- **`Point2D:`** A data structure to hold coordinates of 2D points (and maybe more).
- **`Hierarchy:`** Holds the hierarchy information for CAD entities. These hierarchies can be nested or tree like.
- **`Label:`** A label for a given CAD entity. Note that the label instances should be unique. Whenever you create a new label your library should make sure that it is not repeated.

Your library, at the minimum, should be able used to generate the following two CAD examples:

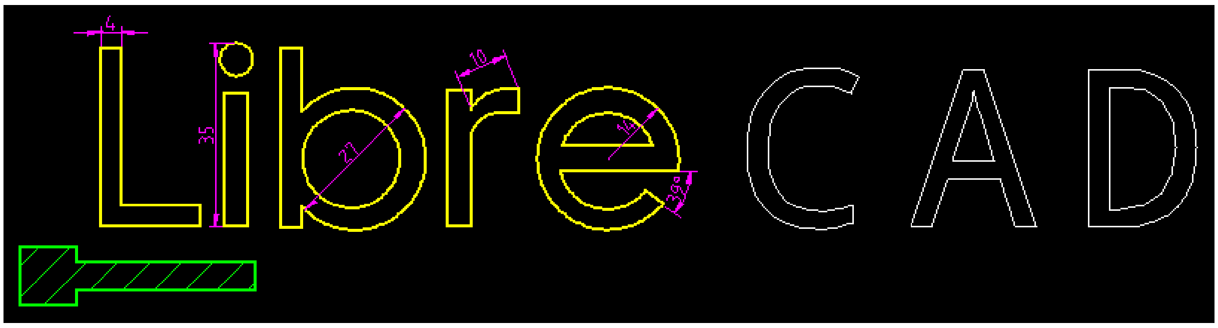

Figure 2 Example CAD.

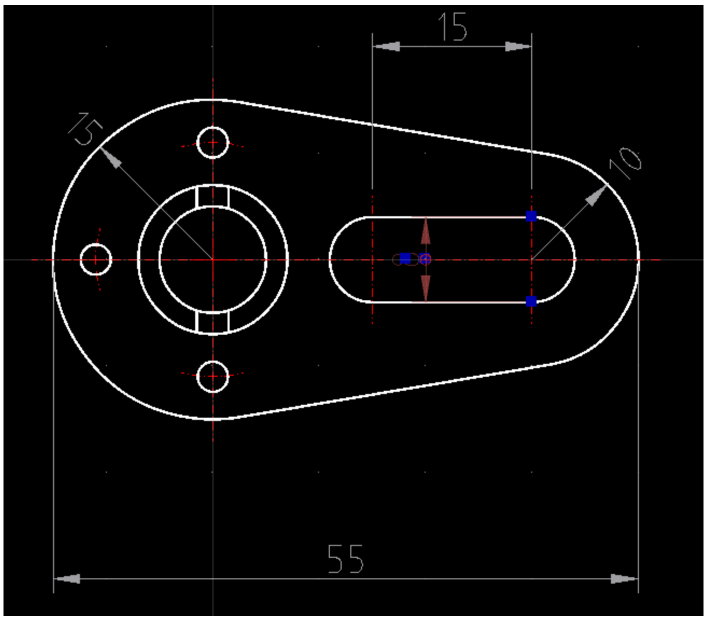

Figure 3 Example CAD.

What to hand in: You are expected to hand in all you source code (library and test programs) along with your makefile in a ZIP or similarly archived filed named "**cse102summerproject_lastname_firstname_studentno.zip**". When the makefile is run, it should compile everything and produce a test program. The test program should illustrate all the above functionality.