LAB TEST – 2 **June 16, 2021**

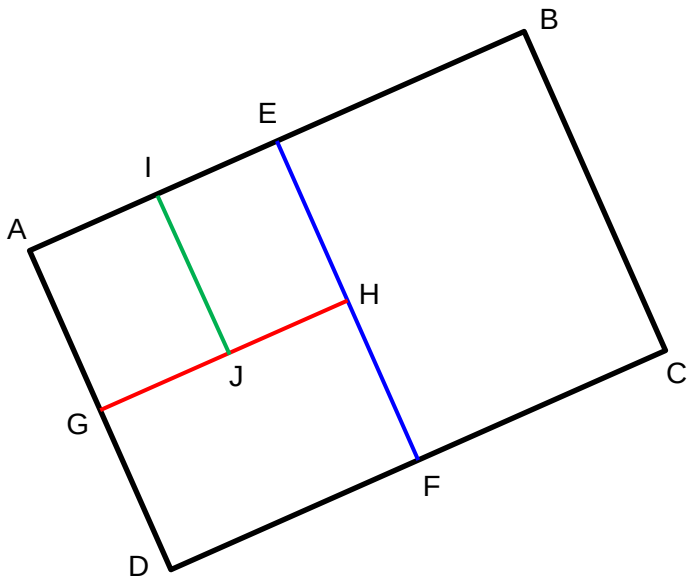
---

1. **The Rectangle Partitioning Problem.** The task here is to recursively partition a given rectangle until the area of each piece is less than a given number, W. Use the following structure for representing rectangles:

```
struct point {
      float x;          // x coordinate of the point
      float y;          // y coordinate of the point
};
struct rect {
      struct point p1;
      struct point p2;
      struct point p3;
      struct point p4;
};
```

where `p1` and `p3` are diagonally opposite corner points of the rectangle. Likewise, `p2` and `p4` are diagonally opposite corner points of the rectangle. The recursive algorithm for partitioning is as follows:

```
Algorithm Partition (Rectangle R, Area Limit W)
begin
      if area of R is less than or equal to W
      then add R into the set of final rectangles ;
      else begin
            Split R into R1 and R2 of equal area by halving R on it's larger side ;
            Partition(R1, W) ;
            Partition(R2, W) ;
      end
end
```



At the first level of recursion the rectangle, ABCD, is split using the blue line, EF. Note that the larger edge, AB, is being split (that is, AB > BC). After the split, in rectangle AEFD, we have EF > FD, and therefore the red line, GH, splits AEFD in the second level of recursion. Now GH > AG, and therefore in the third level of recursion the green line, IJ, splits AGHE. If the area of AIJG is less than or equal to W, then no further splitting will take place. Note that the recursive splitting of EBCF and GHFD has not been shown here, but they too will be split on other branches of recursion.

Multiple splits may happen consecutively on the same direction, though that is not illustrated here.

Write the following in C:

(a) Write the following function to compute the area of a rectangle:

```
float area(struct rect R)
```

(b) Write the following function to split a rectangle R into rectangles R1 and R2 by splitting R on it's larger side:

```
void split(struct rect R, struct rect *R1, struct rect *R2)
```

(c) Write a recursive function, `partition()`, implementing Algorithm Partition (provided above) using the functions, `area()` and `split()`. The function should use an array of structures to record the set of final rectangles (all having area less than or equal to W).

(d) Write a C program which does the following:

(i) It reads the corner points of a rectangle and a floating point number, W. You may assume that the user provides the corner points in clockwise order.

(ii) It uses the function, `partition()`, to find the set of final rectangles.

(iii) It prints the list of final rectangles. In each line of this output it should print the coordinates of all four corners of a rectangle. A sample output could be:

```
RECT1: (0,0) (0,2) (2,0) (2,2)
RECT2: (0,2) (0,4) (2,2) (2,4)
```

and so on …

(iv) It prints the list of final rectangles which do not share a border with the original rectangle. In the figure, all the rectangles share a (black) border with the original rectangle, but one more level of recursion will create rectangles that are strictly inside the original rectangle and do not have any point on the border of the original rectangle.

Some useful formulae:

- Mid-point of (x1, y1) and (x2, y2) is $\left(\frac{x1+x2}{2}, \frac{y1+y2}{2}\right)$
- Distance between (x1, y1) and (x2, y2) is $\sqrt{(y2-y1)^2+(x2-x1)^2}$

[Submission Filename: ⟨Your roll number⟩LT2.c

If your roll number is 20CS30099, then the filename for this task will be 20CS30099LT2.c]