**Algorithm for Midpoint Circle Algorithm (MCA)**

**Inputs:**

* r (radius of the circle)
* xc (x-coordinate of the circle's center)
* yc (y-coordinate of the circle's center)

**Step-by-Step Algorithm:**

1. **Initialize Parameters**:
   * Set x = 0 (start at the center of the circle along the x-axis).
   * Set y = r (start at the topmost point of the circle along the y-axis).
   * Compute the initial decision parameter p = 1 - r.
2. **Create Empty Lists**:
   * xes = [] (empty list to store x-coordinates of circle points).
   * yes = [] (empty list to store y-coordinates of circle points).
3. **Plot Initial Points**:
   * Call plot(xes, yes, x, y, xc, yc) to add the first 8 symmetric points around the circle (using the symmetry of the circle). This function extends xes and yes with the points (x + xc, y + yc), (-x + xc, y + yc), (-x + xc, -y + yc), (x + xc, -y + yc), and their symmetric points for the other quadrants.
4. **Loop to Compute and Plot Points**:
   * **While x < y** (loop until the x-coordinate is less than the y-coordinate):
     + Increment x by 1 (x = x + 1).
   * **Decision Making**:
     + - If p < 0: The next point is horizontally aligned, so update p as p = p + 2\*x + 1.
       - If p >= 0: The next point is diagonally aligned (both horizontally and vertically), so update y as y = y - 1 and update p as p = p + 2\*(x - y) + 1.
   * Call plot(xes, yes, x, y, xc, yc) again to plot the new points after the update.
5. **Display the Circle**:
   * Use plt.scatter(xes, yes, marker='o') to plot all the points stored in xes and yes.
   * Add grid lines with plt.grid(True) for better visualization.
   * Show the plot using plt.show().