(1) Trace the KMP algorithm with pat = ababaabc and text = ababababaababaabc.

(2) The algorithm for intersection in page 54 is corrected as follows:

```c
struct point {
    int x, y;
};
struct segment {
    struct point p1, p2;
};
int turn(struct point q0, struct point q1, struct point q2) {
    int dx1, dx2, dy1, dy2;
    dx1 = q1.x - q0.x; dy1 = q1.y - q0.y;
    dx2 = q2.x - q0.x; dy2 = q2.y - q0.y;
    if (dx1*dy2 > dy1*dx2) return 1; /* case 1 */
    if (dx1*dy2 < dy1*dx2) return -1; /* case 2 */
    if ((dx1*dx2 < 0)||(dy1*dy2<0)) return -1; /* case 3 */
    if ((dx1*dx1+dy1*dy1) < (dx2*dx2+dy2*dy2)) return 1; /* case 4 */
    return 0; /* case 5 */
}
int intersect(struct segment s1, struct segment s2) {
    if (((s1.p1.x==s1.p2.x)&&(s1.p1.y==s1.p2.y))||
        ((s2.p1.x==s2.p2.x)&&(s2.p1.y==s2.p2.y))) /* case 1: One of two segments shrinks to one point */
        return ((turn(s1.p1, s1.p2, s2.p1)*turn(s1.p1,s1.p2,s2.p2))<=0)
        || ((turn(s2.p1, s2.p2, s1.p1)*turn(s2.p1,s2.p2,s1.p2))<=0);
    else /* case 2: Neither of segments is a single point */
        return ((turn(s1.p1, s1.p2, s2.p1)*turn(s1.p1,s1.p2,s2.p2))<=0)
        && ((turn(s2.p1, s2.p2, s1.p1)*turn(s2.p1,s2.p2,s1.p2))<=0); /* case 5 */
}
```

Trace this algorithm for the following sets of points.
Specify co-ordinates for the following and trace.

Specify which case you hit when you go through ‘turn’.

(3) Trace Graham’s algorithm for a convex hull with the following points.

(2,2), (1,1) (5,2), (8,2), (8,8), (4,6), (3,7), (2,8), (6,4), (7,8), (10,4), (1,6)

Name these points p[1], ..., p[12]. Swap the lowest point and p[1] so that p[1] becomes the lowest point. Next sort p[1], ..., p[12] according to their angles. At this stage, draw those points on the x-y plane. Start from P[1] and select successive points according to the algorithm. Give the snapshots at each selection and show whether already selected points are thrown away or not.