In the above figure, the top two lines correspond to the upper boundary of any polygon and the bottom two lines correspond to the lower part. If a vertex is convex, for the bottom two lines, $p_3$ must be above the line passing through $p_1$ and $p_2$. Similarly, for the top two lines, $p_3$ must be below the line passing through $p_1$ and $p_2$. It is very quick to check whether a point is above or below a given line expression. Besides, I could use the function in the previous section to help me figure out the line equation given any two points. By comparing the x-coordinates of $p_1$ and $p_2$, I could also quickly distinguish which case it is.

---

```
for index in all_polygons:
   polygon=all_polygons[index]
   for vertex in range(0,len(polygon)):
      if(vertex==0):
         to_vertex=polygon[1]
         from_vertex=polygon[-1]
      else if(vertex==len(polygon)-1):
         to_vertex=polygon[0]
         from_vertex=polygon[-2]
      else:
         to_vertex=polygon[vertex+1]
         from_vertex=polygon[vertex-1]
      mode_leftright=Checkleft_right(polygon[vertex],to_vertex,from_vertex)
      mode_convexity=CheckConvex(polygon[vertex],to_vertex,from_vertex)
      switch(mode_leftright):
         case 1: #right/right
            if(mode_convexity==convex):
               return mode3
            else:
               return mode4
         case 2: #left/left
            if(mode_convexity==convex):
               return mode1
            else:
               return mode2
         case 3: #left/right
            if(mode_convexity==convex):
               return mode5
            else:
               return mode6
def Checkleft_right(p_m,p_t,p_f):
   if(p_m.x<p_t.x and p_m.x<p_f.x):
      return 2
   else if(p_m.x>p_t.x and p_m.x>p_f.x):
      return 1
   else:
      return 3
def CheckConvex(p_m,,p_f):
   if(p_f.x<p_m.x): #the bottom two lines
```

```
    (a,b,c)=computeLineThroughTwoPoints(p_m,p_f): #from homework1
    y=(-a*p_t[0]-c)/b #b is not 0 since the line is not vertical
    if(y>=p_t[1]):
       return non-convex
    else:
       return convex
 else: #the top two lines
    (a,b,c)=computeLineThroughTwoPoints(p_m,p_f): #from homework1
    y=(-a*p_t[0]-c)/b #b is not 0 since the line is not vertical
    if(y=p_t[1]):
       return non-convex
    else:
       return convex
```

---