**Problem 15 clean campus[Pseudocode]**

[main]

Read number of testcase

Repeat the code below number of testcase

Read number of dots

Read dots

minIdx <- Find left most dot’s index(if dots have same x-coordinate, select the point with smallest y-coordinate)

//find dot that included convexHull

cur <- minIdx

while(true){

convexHullDot[head++] = dot[cur]

std = (cur+1)%numOfDots

//choose points with CCW and largest angle

for i<- 0 to numOfDots-1

while(isCCW(dot[cur], dot[i], dot[std])

std =i

cur = std

if cur same as minIdx (return to the first dot)

break

}

Find a dot connect with dot(0,0)

The dot has a minimum d

(when d1 = distance(stdDot, convexHullDot[i]), D2 = distance(stdDot, convexHullDot[(i+1)%head]), d3 = distance(convexHullDot[(i + 1) % head], convexHullDot[i]), d = d1 + d2 - d3)

result <- calculate distance of convexHull + 2 + d

print result

**Problem 16 a marketing strategy[Pseudocode]**

[main]

Read number of dot

Read dots

Sort dots ascending order based on the x-coordinate

minDis <- closestPair(dots, 0, number of dot-1)

if minDis is bigger than 10000

print infinity

else print minDis

[closestPair(dots, s, l)]

If l-s same as 0

Return 0

Else if l-s same as 1

Return distance between two point dots[s] and dots[l]

Else if l-s same as 2

D1 <- distance between two point dots[s] and dots[s+1]

D2 <- distance between two point dots[s] and dots[s+2]

D3 <- distance between two point dots[s+1] and dots[s+2]

Return min(d1, d2, d3)

midIdx <- (s+e)/2

ds = closestPair(dots, s, midIdx)

dl = closestPair(dos, midIdx+1, l)

minDis <- min(ds, dl)

for i<- s to l

if dots[midIdx].x – minDis <= dos[i].x <= dots[midIdx].x + minDis

S[s\_head++] = dot[i]

Sort S ascending order based on y-coordinate

For i<-0 to s\_head-1

For j<-1 to 7

If j+i>=s\_head then break

D <- distance between two dots S[i] and S[i+k]

If D is smaller than minDis

Then minDis <- D

Return minDis