p1 (1 point)

Diferencias entre un `low-pass filter` y un `high-pass filter`

- `low-pass filter`: conocidos tambien como `smoothing-filter`, son filtros que sirven para suavizar las imagenes, removiendo el ruido, las frecuencias altas y dar efecto blurr a las imagenes
- `high-pass filter`: son filtros que remueven las frecuencias bajas de la señal, dejando intactas las frecuencias altas. Útiles en casos como `edge detection` ya que captura este tipo de detalles de las imagenes.

En resumen un `low-pass filter` remueve frecuencias altas, mientras que un `high-pass filter` remueve frecuencias bajas. Las frecuencias son cambios de intensidad de color en la imagen. Lo que hace que tengan resultados opuestos, `low-pass filter` remueve el detalle, mientras que `high-pass filter` resalta el detalle.

p2 (1 point)

Aplicaciones de modelo de color

- **RGB**: utilizado comunmente para representar imagenes en monitores.
- **HSV**: utlizado en procesamiento de imagenes como paso previo de inputs de modelos de deteccion de objetos o segmentacion semantica.

p3 (1 point)

En el dominio de frecuencias, la operacion de convolucion en el dominio espacial corresponde a la multiplicacion de las transformadas de Fourier de las dos señales implicadas. Sean f y g las señales implicadas y $*$ la operacion de convolucion:

$$F(f * g) = F(f) \cdot F(g)$$

Donde F denota la transformada de Fourier.

p4 (2 point)

Para incrementar el contraste de una imagen, se proponen 2 alternativas:

- `Histogram equalization`: algoritmo que utiliza el histograma y la `CDF` (Cumulative Distribution Function) para aumentar el rango de colores de la imagen.
- `Mapeo a un mayor rango (Naive approach)`: Dada una imagen de 1 canal con un rango de valores de pixeles entre $[a, b]$ lo que se propone es mapear todos los pixeles de este rango a uno mayor $[c, d]$ donde $0 \leq c \leq a - 1$ y $b + 1 \leq d \leq 255$ con una operacion simple de `min-max normalization`, esta propuesta se puede aplicar por cada canal en imagenes RGB.

p5 (2 point)

Dada la tupla RGB (34, 139, 34) se convierte a HSV con los siguientes pasos:

- Normalizacion de los valores RGB:
  - R = 34 / 255 = 0.133
  - G = 139 / 255 = 0.545
  - B = 34 / 255 = 0.133
- Calculo de Max y Min entre los RGB normalizados:
  - Max = max(0.133, 0.545, 0.133) = 0.545
  - Min = min(0.133, 0.545, 0.133) = 0.133
- Calculo del valor V(Value)
  - V = Max
  - V = 0.545
- Calculo del valor S(Saturation)
  - S = (Max - Min) / Max.
  - S = 0.756
- Calculo del valor H(Hue)
  - Dado que Max = G se utiliza la formula
    - H = 60 * ((B - R) / (Max - Min)) + 120
  - Entonces,
    - H = 60 * ((B - R) / (Max - Min)) + 120
    - H = 60 * ((0.133 - 0.133) / (0.545 - 0.133)) + 120
    - H = 120 degrees Por lo tanto el HSV eequivalente al RGB (34, 139, 34) es:
  - H: 120 degrees
  - S: 75.6%
  - V: 54.5%

p6 (2 point)

- `point in segment`
  - Calcular la componente z del producto cruz entre los vectores $\overrightarrow{ap}$ y $\overrightarrow{ab}$.
  - Para que el punto p pertenezca al segmento $\overrightarrow{ap}$ no puede estar a la derecha ni a la izquierda del vector (1, ni -1), por lo tanto la componente z del resultado anterior deberia ser 0.
  - Si la componente z de $\overrightarrow{ap} * \overrightarrow{ab}$ es 0 entonces p es colineal y tiene posibilidades de pertenecer al segmento $\overrightarrow{ab}$.
    - Verificamos esto asegurandonos que las componentes x, y, z pertenezcan a los rangos de $[a_x, b_x], [a_y, b_y], [a_z, b_z]$, respectivamente
    - Si las componentes pertenecen a los rangos, entonces el punto p esta en el segmento. Caso contrario, no esta en el segmento.

p7 (2 point)

Por definicion el modulo del producto cruz $|v_1 * v_2|$ representa el area del paralelogramo.

$|v_1 * v_2| = |x_1y_2 - y_1x_2| = | \det \begin{pmatrix} x_1 & y_1 \ x_2 & y_2 \end{pmatrix}|$

p8 (4 point)

p9 (5 point) Maximo numero de puntos colineales

- Solucion O(N^3):
  - Iteramos sobre todos los posibles pares de puntos
  - Luego para cada par de puntos que forman un segmento, verificamos cuantos puntos son colineales a ese segmento

```
max_number_of_collinear = 2
for point_seed in points:               // N
    for other_point in points:         // N
        if equal(other_point, point_seedd)
            continue
        collinears = 2
        for another_point in points:    // N
            if all_different(another_point, other_point, point_seedd) and \
                another_point is collinear with the segment point_seed-
other_point
                collinears += 1
        max_number_of_collinear = max(max_number_of_collinear, collinears)
return max_number_of_collinear
```

- Solucion O(N^2) sweep line
  - Se utiliza un enfoque sweep line para reducir la cantidad de comparaciones
    - Evento: entra punto
      - En este evento la sweep line intersecta con un punto, lo que se hace luego es verificar si el punto es colineal en O(1) a uno de los end-points de la lista de eventos
      - Los end-points son definidos como puntos finales de alguno de los segmentos formados en los eventos anteriores
      - Estos end-points almacenan la cantidad de puntos colineales a su segmento asociado

p10 (4 point)

- **SOLUCION** Circular Sweep Line:
  - Sea el radar definido por el origen (0, 0)
  - Ordenar los puntos por la distancia del punto mas cercano (radio al origen) y luego por la distancia del punto mas lejano.
  - Utilizamos una "circular sweep line", definida por un radio en la cual puede suceder 2 eventos
    - Ingresa un misil en orbita

            - Cuando sucede esto mantenemos la cuenta de cuantos misiles estan en la orbita, aumentandola
            - Si en algun momento la cantidad de misiles en orbita es > 1, retornamos False
        - Sale un misil de orbita
            - Cuando sucede esto sacamos un misil de la orbita, y reducimos la cuenta de misiles actuales en 1
    - Retornamos verdadero si la sweep line itero sobre todos los puntos y nunca retorno False
- **COMPLEJIDAD** este es un equivalente al Segment Intersection in a Set of Segments el cual puede calcularse en O(nlogn) con el enfoque de Shamos & Hoey.