INF-253 Lenguajes de Programación
Tarea 5: Prolog

Profesor: Roberto Diaz
Ayudante Cátedra: Anastasiia Fedovora
Ayudante Tareas: Héctor Larrañaga - Sebastián Campos - Axel Reyes

1. Mecánica

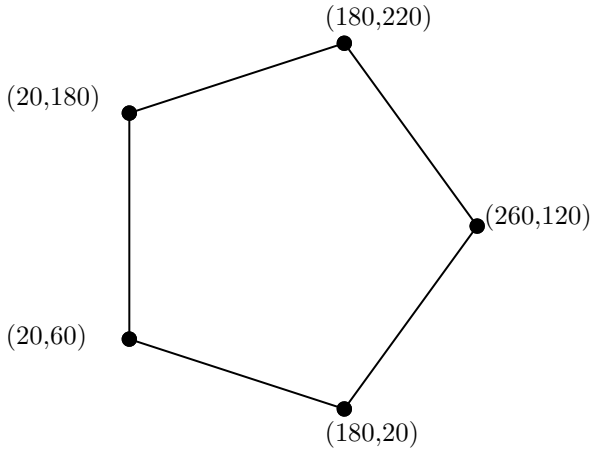
Se les entregarán dos problemas a resolver utilizando el lenguaje de programación Prolog.

Tendrán la libertad de poder realizar su propio formato de consulta, **por lo que se debe especificar el cómo realizarlas en uno o varios comentarios en el código**.

Para poder realizar la tarea, se debe utilizar SW1-Prolog, el cual se puede encontrar en: https://www.swi-prolog.org/download/stable

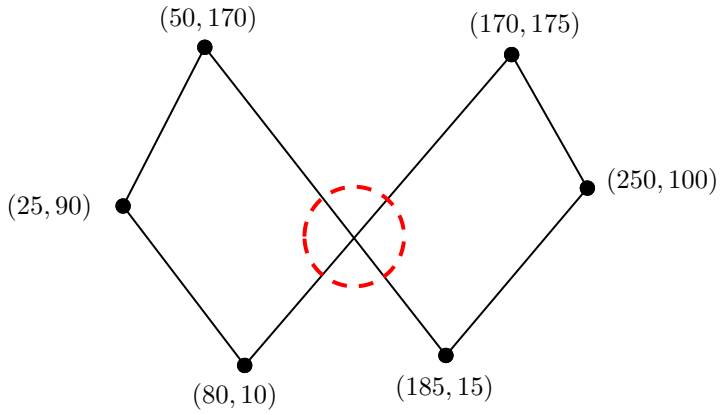
2. Validación de Sectores

Los sectores corresponderán a polígonos simples. El sector será entregado como input en forma de lista de puntos, donde los puntos se verán representados como listas de dos elementos, las cuales serán las coordenadas x e y del punto en el plano 2D. A continuación se muestran ejemplos de sectores válidos e inválidos:




```
sector([[20,180],
 [20,60],
 [180,20],
 [260,120],
 [180,220],
 [20,180]]
).
```

Figura 1: Definición de un sector válido




```
sector([[50,170],
 [185,15],
 [250,100],
 [170,175],
 [80,10],
 [25,90],
 [50,170]]
).
```

Figura 2: Sector inválido

Tendrán que crear una regla que verifique si su componente es un sector válido o no. Ejemplo:

```
?- sector([[20,180],[20,60],[180,20],[260,120],[180,220],[20,180]])
true.

?- sector([[50,170],[185,15],[250,100],[170,175],[80,10],[25,90],[50,170]])
false.
```

Asuma que los sectores entregados tendrán siempre como mínimo 3 puntos, por lo que las listas de puntos siempre tendrán mínimo 4 elementos (ya que el primer y último punto se repiten para cerrar la figura). Asuma también que un punto no se repetirá a menos que sea el primero y el último.

3. Detección de adyacencia

Usando la definición de sector descrita anteriormente, se les pedirá verificar si dos sectores distintos no superpuestos son adyacentes entre si. Dos sectores serán adyacentes si y sólo si comparten un segmento como mínimo.

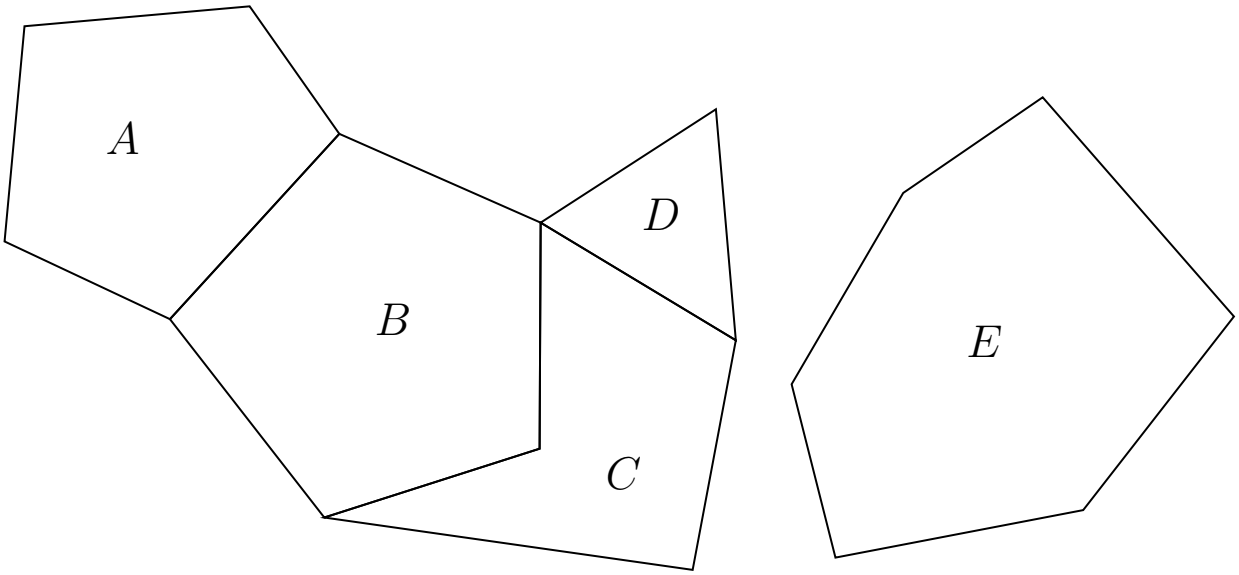


Figura 3: Ejemplo de adyacencia de sectores.

En el ejemplo de la Figura 3, el sector B es adyacente tanto a A como a C, pero no a D. Este último es adyacente solamente con el sector C. El sector E no es adyacente a ningún otro sector.

Para este problema, se debe implementar una regla llamada `ady` que verifique si dos sectores son adyacentes.

Ejemplo:

```
?- ady([[1,1],[2,2],[3,1],[1,1]],[[10,1],[12,1],[11,2],[10,1]])
false

?- ady([[1,1],[2,2],[3,1],[1,1]],[[2,2],[3,1],[4,5],[2,2]])
true
```

4. Reglas del juego

4.1. Archivos a entregar

Se deben implementar ambas soluciones en un archivo llamado sectores.pl.

4.2. Otras Reglas

- La tarea debe tener cada regla comentada siguiendo el siguiente formato:
  /*
  -Descripción parámetro 1: ....
  -Descripción parámetro 2: ,,,,
  .
  .
  .

  Descripción de la regla. */
- Se debe trabajar de forma individual.
- Cuidado con el orden y la identación de su tarea, puede llevar descuentos.
- La copia implica 0.
- El código debe ser correctamente comentado, sino habrá descuento.
- **La entrega debe realizarse en .zip y debe llevar el nombre: Tarea5LP_RolIntegrante-1.zip**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones para la utilización de su programa en caso de ser necesarias.**
- **El no cumplir con las reglas de entrega o formato conllevará un descuento de máximo 30 puntos en su tarea.**
- La entrega será vía aula y el plazo máximo de entrega es hasta el **día 6 de Enero a las 23:55 hrs.**.
- Sólo se contestarán dudas **hasta 3 días** antes de la fecha de entrega original.
- **Habrá un descuento de 20 puntos por cada hora de atraso.**
- Cualquier archivo que falte implica un 0 en el problema.

5. Calificación

- Identificación de sectores válidos (70 pts)
- Adyacencia de sectores (30 pts)