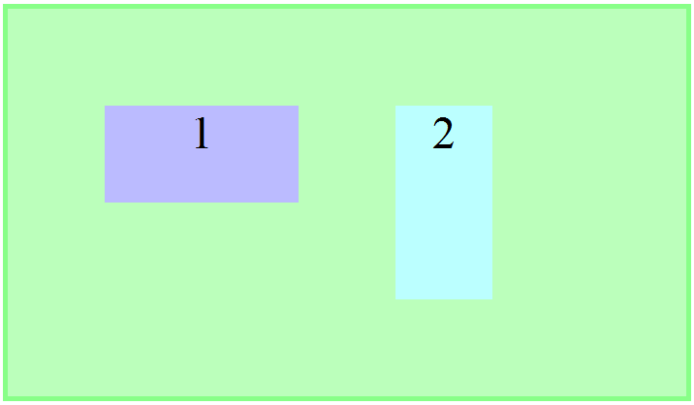
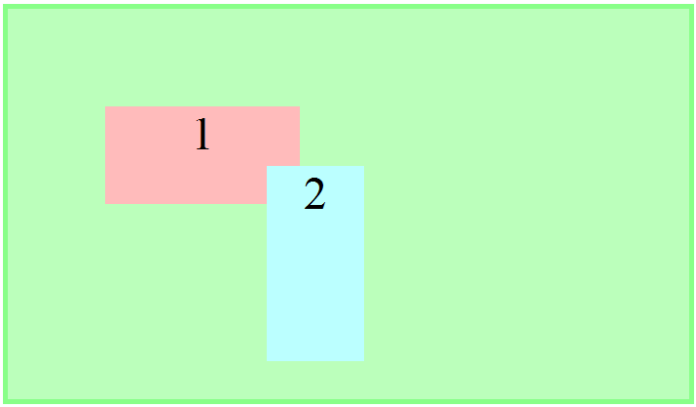
==========

ASSIGNATURA: *Laboratori Multimèdia (GEIGSI)*

==========

(2.5 punts) **Final: 29 de juny de 2016**. Escriviu el codi javascript+jQuery d'un programa que permeti moure el rectangle 2 i quan aquest col·lisiona amb el rectangle 1, aquest canvia el color de background a color "#fbb" (vermell) i quan deixa de col·lisionar torna al seu color "#bbf" (blau).






```
<!DOCTYPE html>
<html><head>
<meta charset="UTF-8">
<title>Pregunta 2. Col·lisió de dos rectangles</title>
<style>
#pista{
        position:relative;
        width:700px; height:400px;
        background-color: #bfb;
        border: thick solid #8f8;
}
.rectangle{
        position:absolute;
        font-size: 3em;
        text-align: center;
}
#rectangle1{
        width:200px; height:100px;
        left:100px; top:100px;
        background-color: #bbf;
}
#rectangle2{
        width:100px; height:200px;
        left:400px; top:100px;
        background-color: #bff;
}
</style></head>
<body>
<div id="pista">
  <div id="rectangle1" class="rectangle">1</div>
  <div id="rectangle2" class="rectangle">2</div>
</div>
<script src="javascript/jquery-1.11.2.min.js"></script>
<script src="javascript/jquery-ui.min.js"></script>
<script>
var pista={w:0, h:0};                  // mides de la pista
var r1={x:0, y:0, w:0, h:0};   // posició i mida
var r2={x:0, y:0, w:0, h:0};   // posició i mida
$(document).ready(function(){
  $("#rectangle2").draggable();  // jQuery UI, fa que el rectangle 2 es pugui arrossegar per l'usuari

  // llegim les mides de la pista i dels rectangles i la posició del rectangle1
```

(1 punt)

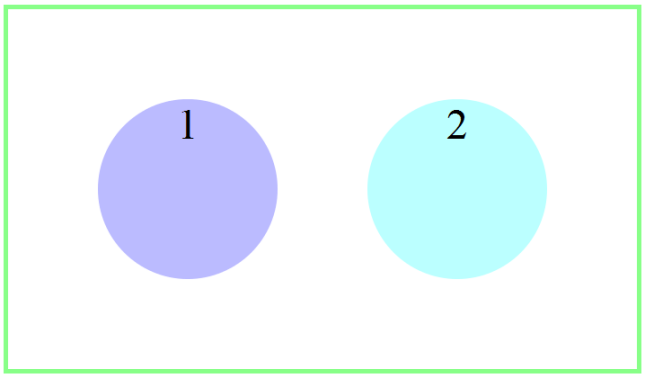
```
pista.w=parseInt($("#pista").css("width"));
pista.h=parseInt($("#pista").css("height"));
r1.x=parseInt($("#rectangle1").css("left"));
r1.y=parseInt($("#rectangle1").css("top"));
r1.w=parseInt($("#rectangle1").css("width"));
r1.h=parseInt($("#rectangle1").css("height"));
r2.w=parseInt($("#rectangle2").css("width"));
r2.h=parseInt($("#rectangle2").css("height"));
```

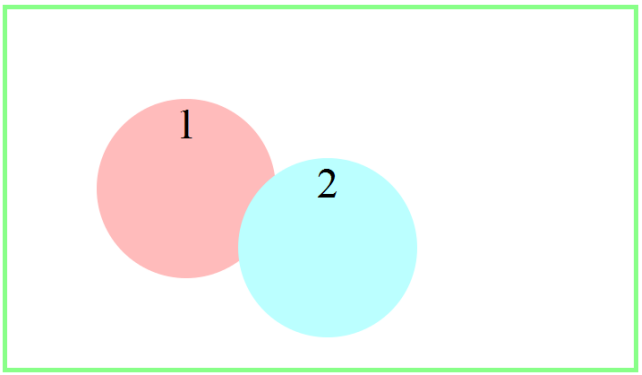
```
  $("#rectangle2").mousemove(function(){
```

(1,5 punt)

```
    function colisioRectRect(r1,r2){
        return !(r2.y+r2.h < r1.y || r2.x > r1.x+r1.w || r2.y > r1.y+r1.h || r2.x+r2.w < r1.x);
    }
    // llegim la posició actual del rectangle 2
    r2.x=parseInt($("#rectangle2").css("left"));
    r2.y=parseInt($("#rectangle2").css("top"));
    //mirem si els dos rectangles col·lisionen
    if(colisioRectRect(r1,r2)){
        $("#rectangle1").css("background-color","#fbb");
    }
    else{
        $("#rectangle1").css("background-color","#bbf");
    }
 });
});
</script></body></html>
```

(2.5 punts) **Recuperació: 13 de juliol de 2016**. Escriviu el codi javascript+jQuery d'un programa que permeti moure la bola 2 i quan aquesta col·lisiona amb la bola 1, aquesta darrera canvia el color de background a color "#fbb" (vermell) i quan deixa de col·lisionar torna al seu color "#bbf" (blau).






```
<!DOCTYPE html>
<html><head>
<meta charset="UTF-8">
<title>Pregunta 2. Col·lisió de dues boles</title>
<style>
#pista{
        position:relative;
        width:700px; height:400px;
        border: thick solid #8f8;
}
.bola{
        position:absolute;
        width:200px; height:200px; border-radius: 100px;
        font-size: 3em; text-align: center;
}
#bola1{
        left:100px; top:100px;
        background-color: #bbf;
}
#bola2{
        left:400px; top:100px;
        background-color: #bff;
}
</style>
</head>
<body>
<div id="pista">
  <div id="bola1" class="bola">1</div>
  <div id="bola2" class="bola">2</div>
</div>

<script src="javascript/jquery-1.11.2.min.js"></script>
<script src="javascript/jquery-ui.min.js"></script>
<script>
// objectes
var pista={w:0, h:0};       // mides de la pista
var bola1={x:0, y:0,        // posició
           w:0, h:0};       // mida
var bola2={x:0, y:0,        // posició
           w:0, h:0};       // mida
$(document).ready(function(){
  $("#bola2").draggable();  // jQuery UI, fa que el rectangle 2 es pugui arrossegar per l'usuari

  // llegim les mides de la pista i de les boles i la posició de la bola1
```

(1 punt)

```
  pista.w=parseInt($("#pista").css("width"));
  pista.h=parseInt($("#pista").css("height"));
  bola1.x=parseInt($("#bola1").css("left"));
  bola1.y=parseInt($("#bola1").css("top"));
  bola1.w=bola1.h=bola2.w=bola2.h=parseInt($("#bola1").css("width"));
```

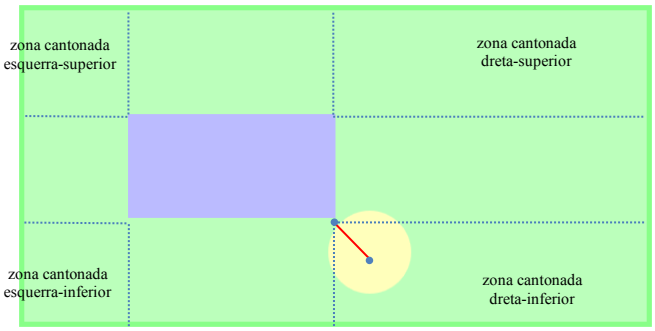
```
  $("#bola2").mousemove(function(){
```

(1,5 punt)

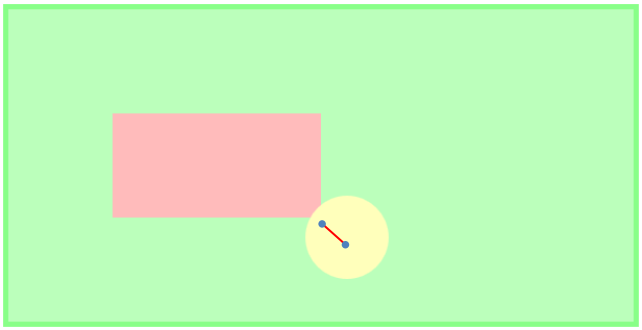
```
   function colisioBolaBola(b1,b2){
    return distancia({x:b1.x+b1.w/2,y:b1.y+b1.h/2},{x:b2.x+b2.w/2,y:b2.y+b2.h/2})<b1.w;
   }
   function distancia(p1,p2){  // p1, p2={x:, y:}
    return Math.sqrt(Math.pow(p2.x-p1.x,2)+Math.pow(p2.y-p1.y,2));
   }
   // llegim la posició actual de la bola 2
   bola2.x=parseInt($("#bola2").css("left"));
   bola2.y=parseInt($("#bola2").css("top"));
   //mirem si les dues boles col·lisionen
   if(colisioBolaBola(bola1,bola2)){
     $("#bola1").css("background-color","#fbb");
   }else{
     $("#bola1").css("background-color","#bbf");
   }
  });
});
</script>
</body></html>
```

(3 punts) **Final: 27 de juny de 2017.** Escriviu el codi javascript+jQuery d'un programa que permeti moure el cercle i quan aquest col·lisiona amb el rectangle, aquest canvia el color de background a color "#fbb" (vermell) i quan deixa de col·lisionar torna al seu color "#bbf" (blau). Observeu que la bola no col·lisiona amb el rectangle si el quadrat que la conté no col·lisiona amb el rectangle. Per altra banda, si el centre de la bola està en alguna de les zones cantonada, no hi haurà col·lisió si la distància del centre de la bola al punt de la cantonada corresponent és més gran que el radi de la bola (observeu la imatge).

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Col·lisio bola-rectangle</title>
<style>
#pista{
  position:relative;
  width:600px; height:300px;
  background-color: #bfb;
  border: thick solid #8f8;
}
#bola{
  position:absolute;
  width:80px; height:80px; border-radius: 40px;
  left:10px; top:10px;
  background-color: #ffb;   z-index: 100;
}
#rectangle{
  position:absolute;
  width:200px; height:100px;
  left:100px; top:100px;
  background-color: #bbf;
}
</style>
</head>
<body>
<div id="pista">
  <div id="bola"></div>
  <div id="rectangle"></div>
</div>
<script src="javascript/jquery-1.11.2.min.js"></script>
<script src="javascript/jquery-ui.min.js"></script>
<script>
// objectes
var pista={w:0, h:0};
var bola={x:0, y:0, w:0, h:0};
var rectangle={x:0, y:0, w:0, h:0};

$(document).ready(function(){
  // llegim l'estat inicial dels objectes: bola, pista i rectangle
```



No hi ha col·lisió: distància > radi



Hi ha col·lisió: distància ≤ radi

(1 punt)

```
  bola.w=parseInt($("#bola").css("width"));
  bola.h=parseInt($("#bola").css("height"));
  pista.w=parseInt($("#pista").css("width"));
  pista.h=parseInt($("#pista").css("height"));
  rectangle.w=parseInt($("#rectangle").css("width"));
  rectangle.h=parseInt($("#rectangle").css("height"));
  rectangle.x=parseInt($("#rectangle").css("left"));
  rectangle.y=parseInt($("#rectangle").css("top"));
```

```
  $("#bola").draggable();  // jQuery UI, fa que la bola es pugui arrossegar per l'usuari
  $("#bola").mousemove(function(e){moure();}); // cada vegada que es mou la bola, s'executa la funció moure()
});

function moure(){
  // llegim la posició actual de la bola
  bola.x=parseInt($("#bola").css("left"));
  bola.y=parseInt($("#bola").css("top"));
  //mirem si la bola toca alguna vora del rectangle
  if(colisioRectBola(rectangle,bola)){
    $("#rectangle").css("background-color","#fbb");
  }else{
    $("#rectangle").css("background-color","#bbf");
  }
}

// determina si els dos rectangles r1 i r2 col·lisionen
function colisioRectRect(r1,r2){  // r1, r2={x:, y:, w: , h:}
```

(1 punt)

```
  return !(r2.y+r2.h < r1.y || r2.x > r1.x+r1.w || r2.y > r1.y+r1.h || r2.x+r2.w < r1.x);
```

```
}
```

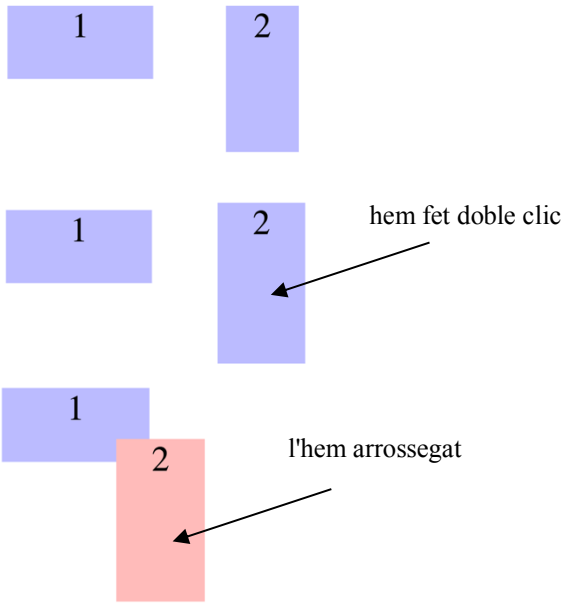
```
// determina si el rectangles r i la bola b col·lisionen
function colisioRectBola(r,b){ // r, b={x:, y:, w: , h:}
  if(!colisioRectRect(r,b)){
    return false;
  }
  else{ // hem de mirar les quatre zones cantonada
```

(1 punt)

```
    var pEsqSupRect={x:r.x, y:r.y},             pDreSupRect={x:r.x+r.w, y:r.y},
        pEsqInfRect={x:r.x, y:r.y+r.h},         pDreInfRect={x:r.x+r.w, y:r.y+r.h};
    var centreBola={x:b.x+b.w/2, y:b.y+b.h/2};
    if(centreBola.x<r.x && centreBola.y<r.y){
        return distancia(centreBola,pEsqSupRect)<b.w/2;
    }
    else if(centreBola.x>r.x+r.w && centreBola.y<r.y){
        return distancia(centreBola,pDreSupRect)<b.w/2;
    }
    else if(centreBola.x<r.x && centreBola.y>r.y+r.h){
        return distancia(centreBola,pEsqInfRect)<b.w/2;
    }
    else if(centreBola.x>r.x+r.w && centreBola.y>r.y+r.h){
        return distancia(centreBola,pDreInfRect)<b.w/2;
    }
    else{
        return true;
    }
  }
}

// calcula la distancia entre els punts p1 i p2
function distancia(p1,p2){  // p1, p2={x:, y:}
  return Math.sqrt(Math.pow(p2.x-p1.x,2)+Math.pow(p2.y-p1.y,2));
}

</script>
</body>
</html>
```

(3 punts) **Recuperació: 10 de juliol de 2019.** Escriviu el codi javascript+jQuery d'un programa que permet moure el rectangle 2 amb els següents comportaments:

a. (1 punt) Quan es fa doble clic en un rectangle qualsevol, aquest s'engreixa 20 píxels, és a dir, es fa 20 píxels més ampla i 20 píxels més alt, respecta el centre del rectangle.
b. (2 punts) Quan en moure els rectangle 2, aquest col·lideix amb el rectangle 1, el rectangle 2 canvia el color de background a color "#fbb" (vermell) i quan deixa de col·lidir torna al seu color "#bbf" (blau).




```
<!DOCTYPE html>
<html> <head>
<meta charset="UTF-8">
<title>Col·lisió de dos rectangles</title>
<style>
.rectangle{
        position:absolute;
        font-size: 3em;
        text-align: center;
        background-color: #bbf;
}
#rectangle1{
        width:200px; height:100px;
        left:100px; top:100px;
}
#rectangle2{
        width:100px; height:200px;
        left:400px; top:100px;
}
</style></head>
<body>
  <div id="rectangle1" class="rectangle">1</div>
  <div id="rectangle2" class="rectangle">2</div>

<script src="javascript/jquery-1.11.2.min.js"></script>
<script src="javascript/jquery-ui.min.js"></script>
<script>
$(document).ready(function(){
  $("#rectangle2").draggable();  // jQuery UI, fa que el rectangle 2 es pugui arrossegar per l'usuari

  // apartat a.
```

(1 punt)

```
    $(".rectangle").on("dblclick",function(){
            // objectes
            var $r=$(this);
            var r={x:parseInt($r.css("left")), y:parseInt($r.css("top")),   // posició actual
                   w:$r.width(), h:$r.height()};        // mida actual

            $r.css({
                    top: (r.y-10)+"px", left:(r.x-10)+"px",    // nova posició
                    width: (r.w+20)+"px", height:(r.h+20)+"px"    // nova mida
            })
     });
```

```
  // apartat b.
```

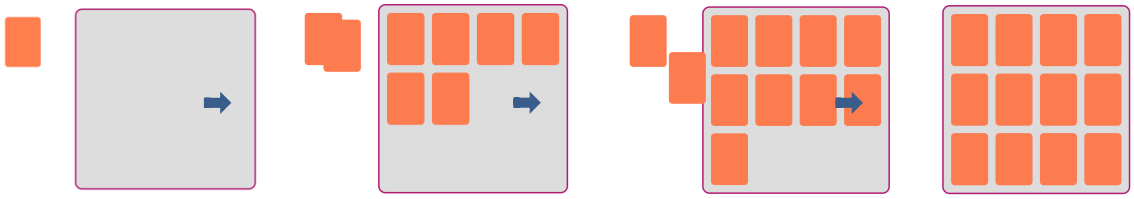
(2 punts)

```
   $("#rectangle2").on("mousemove",function(){
            // objectes
            var r1={x:0, y:0,       // posició
                    w:0, h:0};      // mida
            var r2={x:0, y:0,       // posició
                    w:0, h:0};      // mida
            // llegim les mides i la posició dels rectangles
            var $r=$("#rectangle1");
            r1.x=parseInt($r.css("left")); r1.y=parseInt($r.css("top"));
            r1.w=$r.width(); r1.h=$r.height();
            $r=$("#rectangle2");  // és el que es mou
            r2.x=parseInt($r.css("left")); r2.y=parseInt($r.css("top"));
            r2.w=$r.width();        r2.h=$r.height();

           //mirem si els dos rectangles col·lideixen
            if(!(r2.y+r2.h < r1.y || r2.x > r1.x+r1.w || r2.y > r1.y+r1.h || r2.x+r2.w < r1.x)){
                    $r.css("background-color","#fbb");
            }else{
                    $r.css("background-color","#bbf");
            }
     });
});
</script>
</body></html>
```

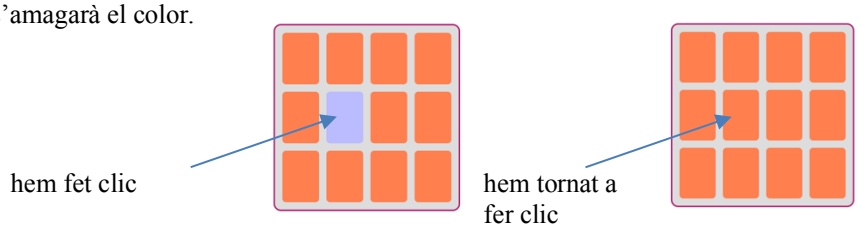
(3 punts) **Final: 25 de juny de 2019.** Completeu el codi javascript+jQuery d'un programa que permeti mostrar un conjunt de 3x4 cartes aparellades per color (hi ha 6 colors diferents), amb la següent funcionalitat:

c. (1 punt) Quan comença el programa, es creen les `nFiles x nColumnes` cartes HTML i es posen en el tauler. La carta HTML de la fila 2 i columna 3 és `<div class="carta" id="f2c3"></div>`

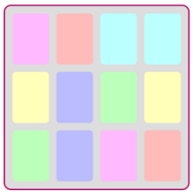
d. (1 punt) Un cop creades les cartes, aniran a la seva posició al tauler amb una animació seqüencial, és a dir, primer anirà la carta `#f1c1` amb una animació de 0.5 segons, després la `#f1c2` amb una animació de 0.5 segons, i així fins al final.



e. (0.5 punt) En fer clic en una carta qualsevol, aquesta mostrarà el seu color. En tornar a fer clic en la mateixa carta, s'amagarà el color.



f. (0.5 punts) En fer doble clic al tauler, es mostraran tots els colors de les cartes



```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Pregunta 1</title>
  <style>
    #tauler{
      position: relative;
      background-color: #ddd;
      border: solid 2px #B01F70; border-radius: 10px;
      margin: 10px auto;
    }
    .carta{
      position: absolute; top: 10px; left: -100px;
      background-color: coral;  /* color del darrera  */
      width: 50px; height: 70px; border-radius: 5px;
    }
    .carta1{background-color: #fbb;}  /*color del davant */
    .carta2{background-color: #bfb;}
    .carta3{background-color: #bbf;}
    .carta4{background-color: #ffb;}
    .carta5{background-color: #bff;}
    .carta6{background-color: #fbf;}
  </style>
</head>
<body>
<div id="tauler">
</div>
  <script src="javascript/jquery-1.11.2.min.js"></script>
  <script>
    $(document).ready(function(){
      var nFiles=3, nColumnes=4;
      var dX=10, dY=10;
      var jocCartes=["carta1","carta2","carta3","carta4","carta5","carta6",
                     "carta1","carta2","carta3","carta4","carta5","carta6"];  // color de la carta
      jocCartes.sort(function(){return Math.random()-0.5});  // barregem el joc de cartes

  //TODO apartat a) creem les cartes
```

1 punt

```
    var s="";
    for(let f=1; f<=nFiles; f++)
      for(let c=1; c<=nColumnes; c++){
        s+='<div class="carta" id="f'+f+'c'+c+'"></div>';
        //s+=`<div class="carta" id="f${f}c${c}"></div>`;  // ES6
      }
    $("#tauler").html(s);
```

```
  // assignem un color a cada carta
  $(".carta").each(function(){
     $(this).data("color",jocCartes.pop());
  });

  // mides del tauler
  var carta=$(".carta");
  var wCarta=carta.width(); var hCarta=carta.height();  // mides de la carta
  $("#tauler").css({
     width: (wCarta+dX)*nColumnes+dX+"px",
    height: (hCarta+dY)*nFiles+dY+"px"
  });

  //TODO apartat b) col·loquem totes les cartes amb una animació seqüencial
```

1 punt

```
  // var i=0
  $(".carta").each(function(i){ // aprofitem l'índex del each com a var i
    var id,f,c;
    carta=$(this);
    id=carta.attr("id");
    f=id.charAt(1); c=id.charAt(3);
    //i++;
    carta.delay(i*500).animate({
      left: (wCarta+dX)*(c-1)+dX+"px",
       top: (hCarta+dY)*(f-1)+dY+"px"
    },500);
  });

  //TODO apartat c) en fer clic a una carta qualsevol, aquesta mostra o amaga el seu color
```

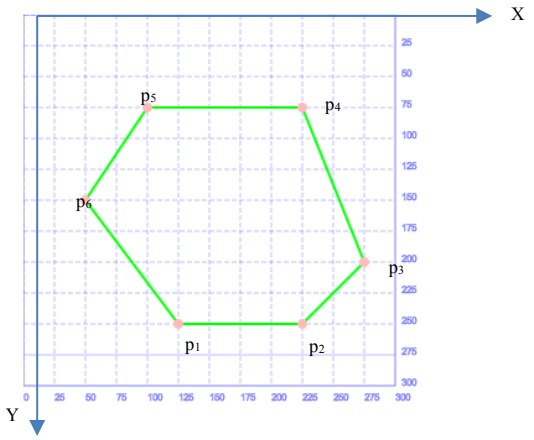
0.5 punts

```
  $(".carta").on("click",function(){
    $(this).toggleClass($(this).data("color"));  // mostrem o amagem el color de la carta
    /* // una altra solució
    var aquesta=$(this);
    if(aquesta.hasClass(aquesta.data("color"))){
      aquesta.removeClass(aquesta.data("color"));
    }
    else{
      aquesta.addClass(aquesta.data("color"));
    }
    */
  });

  //TODO apartat d) en fer doble clic al tauler, es mostren els colors de totes les cartes
```

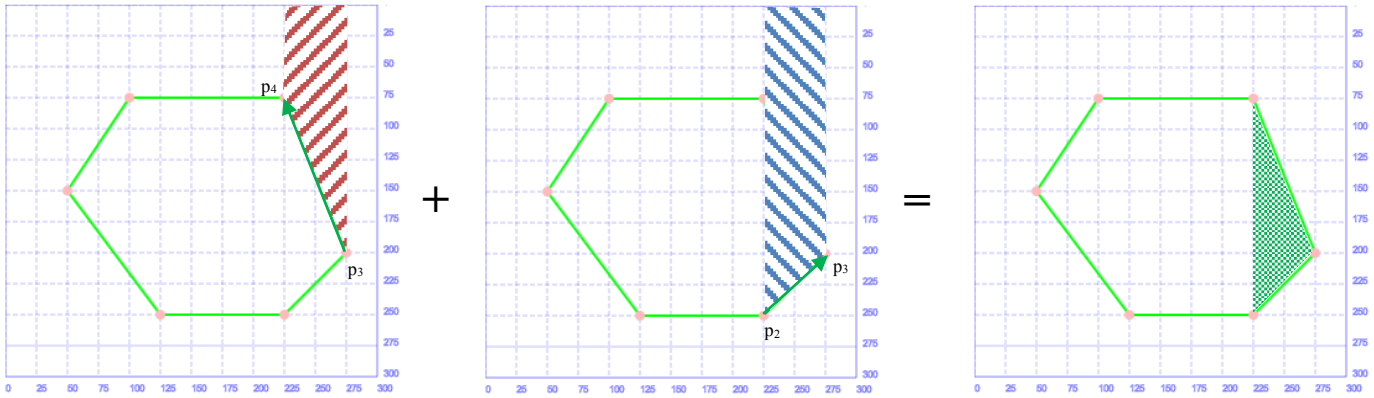
0.5 punts

```
  $("#tauler").on("dblclick",function(){
     $(".carta").each(function(){              // mostrem el color de totes les cartes
       $(this).addClass($(this).data("color"));
     });
  });


  })
</script>
</body>
</html>
```

(3 punts) **Final: 27 de juny de 2017.** Escriviu el codi javascript per a la classe d'objectes Polígon. Un Poligon és una figura plana tancada que està caracteritzada per una seqüència de punts que determinen els segments consecutius que conformen el Polígon (vegeu la figura).



Es defineix el perímetre del Polígon com la suma de les longituds dels segments que el conformen. Un altre paràmetre del Polígon és la seva àrea. Per calcular l'àrea cal anar sumant l'àrea (positiva o negativa) entre el segment (amb direcció) i l'eix X,



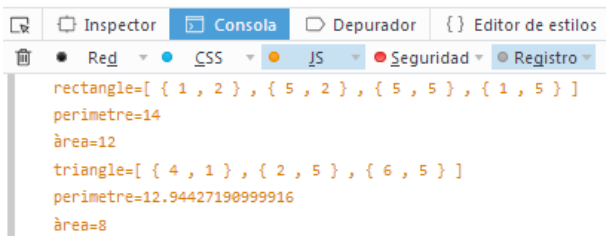
àrea negativa àrea positiva

$$àrea_{p_3 \to p_4} = \frac{1}{2}(p_{3y} + p_{4y})(p_{4x} - p_{3x}) \qquad àrea_{p_2 \to p_3} = \frac{1}{2}(p_{2y} + p_{3y})(p_{3x} - p_{2x})$$

Observeu que si calculem la seqüència d'àrees dels segments fent un recorregut en sentit antihorari, obtindrem l'àrea del Polígon.

Escriviu la classe javascript Poligon que permeti executar el següent codi javascript amb el resultat indicat en la imatge:

```
var rectangle=new Poligon();
rectangle.afegirPunt(new Punt(1,2));
rectangle.afegirPunt(new Punt(5,2));
rectangle.afegirPunt(new Punt(5,5));
rectangle.afegirPunt(new Punt(1,5));
console.log("rectangle="+rectangle);
console.log("perimetre="+rectangle.perimetre());
console.log("àrea="+rectangle.area());

var triangle=new Poligon(false);
triangle.afegirPunt(new Punt(4,1));
triangle.afegirPunt(new Punt(2,5));
triangle.afegirPunt(new Punt(6,5));
console.log("triangle="+triangle);
console.log("perimetre="+triangle.perimetre());
console.log("àrea="+triangle.area());
```



```
class Punt{
  constructor(x,y){
    this.x=x;
    this.y=y;
  }
  ////////////// Mètodes públics /////////////////
  toString(){
    return " { "+this.x+" , "+this.y+" } ";
  }
  ////////////// Mètodes estàtics /////////////////
  static distancia(p1,p2){
    // retorna la distancia entre els dos punts
    return Math.sqrt(Math.pow(p2.x-p1.x,2) + Math.pow(p2.y-p1.y,2));
  }
}

class Poligon{
  constructor(sentitHorari){
    if(sentitHorari!=undefined)
      this.sentitHorari=sentitHorari;
    else
      this.sentitHorari=true; // per defecte, true
    this.punts=[];  // seqüència de punts
  }
  ////////////// Mètodes públics /////////////////
  quantsPunts(){
    return this.punts.length;
  }
  afegirPunt(p){
```

(0.5 punts)

```
              this.punts.push(p);
```

```
  }
  perimetre(){
```

(1 punt)

```
              var perimetre=0;
              for(var i=0; i<this.punts.length-1; i++){
                      perimetre+=Punt.distancia(this.punts[i],this.punts[i+1]);
              }
              perimetre+=Punt.distancia(this.punts[this.punts.length-1],this.punts[0]);
              return perimetre;
```

```
  }
  area(){
```

(1 punt)

```
              function areaSegment(p1,p2){
                      // funció interna que calcula l'àrea sota un segment
                      return (p1.y+p2.y)*(p2.x-p1.x)/2;
              }
              var area=0;
              for(var i=0; i<this.punts.length-1; i++){
                      area+=areaSegment(this.punts[i],this.punts[i+1]);
              }
              area+=areaSegment(this.punts[this.punts.length-1],this.punts[0]);
              if(this.sentitHorari)
                return -area;
              else
                return area;
```

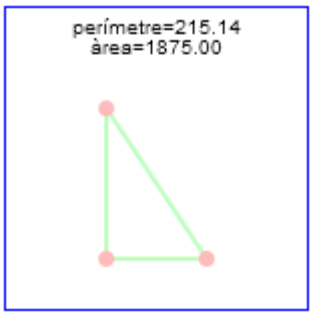
```
  }
  toString(){
```

(0.5 punts)

```
              return "["+this.punts.toString()+"]";
```

```
  }
}
```

(2 punts) **Final: 27 de juny de 2017.** Escriviu una "ampliació" del Poligon de la pregunta anterior perquè es dibuixi en un canvas. Observeu el resultat que s'obté després d'executar el codi que s'indica:




```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Poligon en el canvas</title>
<style>
canvas{
        display:block;
        border:thin #00F solid;
        margin: 0 auto;
}
</style>
</head>
<body>
<canvas id="canvas" width="150" height="150" onclick="draw();" />
<script src="javascript/poligon.js"></script> <!-- codi de la pregunta anterior -->
<script>
class PuntCanvas extends Punt {
```

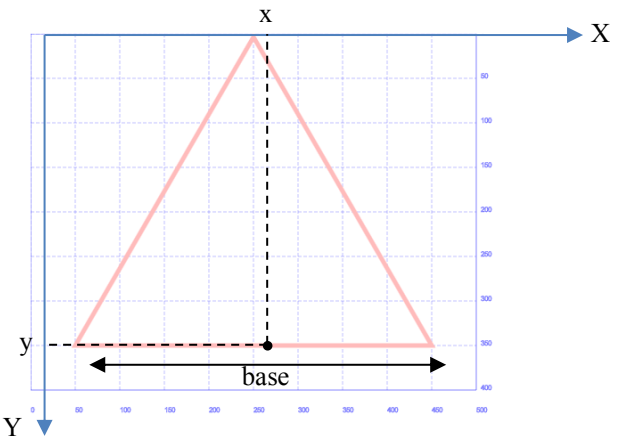
(1 punt)

```
  constructor(x, y, ctx) {
    super(x, y);
    this.ctx = ctx;
  }
  draw() {
    this.ctx.save();
    this.ctx.beginPath();
    this.ctx.fillStyle = "#fbb";
    this.ctx.arc(this.x, this.y, 4, 0, 2 * Math.PI, true);
    this.ctx.fill();
    this.ctx.restore();
  }
}

class PoligonCanvas extends Poligon {
```

(1 punt)

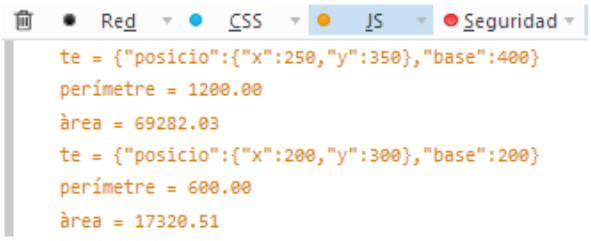
```
  constructor(sentitHorari, ctx) {
    super(sentitHorari);
    this.ctx = ctx;
  }
  draw() {
    this.ctx.save();
    this.ctx.lineWidth = 2;
    this.ctx.strokeStyle = "#bfb";
    this.ctx.beginPath();
    this.ctx.moveTo(this.punts[0].x, this.punts[0].y);
    for (var i = 1; i < this.punts.length; i++) {
      this.ctx.lineTo(this.punts[i].x, this.punts[i].y);
    }
    this.ctx.closePath();
    this.ctx.stroke();
    for (var i = 0; i < this.punts.length; i++) {
      this.punts[i].draw();
    }
    // el text
    this.ctx.font = '10px sans-serif';
    this.ctx.textAlign = "center";
    this.ctx.textBaseline = "middle";
    this.ctx.fillText("perímetre=" + this.perimetre().toFixed(2), this.ctx.canvas.width / 2, 10);
    this.ctx.fillText("àrea=" + this.area().toFixed(2), this.ctx.canvas.width / 2, 20);
    this.ctx.restore();
  }
}

var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

var triangle=new PoligonCanvas(false,ctx);
triangle.afegirPunt(new PuntCanvas(50,50,ctx));
triangle.afegirPunt(new PuntCanvas(50,125,ctx));
triangle.afegirPunt(new PuntCanvas(100,125,ctx));

triangle.draw();

</script>
</body>
</html>
```

(2 punts) **14 de juliol de 2017.** Escriviu el codi javascript per a la classe d'objectes `TriangleEquilater`. Un `TriangleEquilater` té els tres costats iguals. Sempre representarem el triangle amb la base segons l'eix X. Amb aquest conveni, podem caracteritzat el `TriangleEquilater` per la seva posició (punt mig de la base) i la seva mida (mida de la base). Vegeu la imatge següent:



Escriviu la classe javascript `TriangleEquilater` que permeti executar el següent codi javascript amb el resultat indicat en la imatge:

```
var te=new TriangleEquilater(new Punt(250,350),400);
console.log("te = "+te);
console.log("perímetre = "+te.perimetre().toFixed(2));
console.log("àrea = "+te.area().toFixed(2));
te.novaPosicio(new Punt(100,200)).zoom(1/2).novaPosicio(new Punt(200,300)) ;
// hem canviat la posició, el reduim a la meitat i el tornem a canviar de posició
console.log("te = "+te);
console.log("perímetre = "+te.perimetre().toFixed(2));
console.log("àrea = "+te.area().toFixed(2));
```



Nota 1: `(3.45678).toFixed(2)== "3.46"` és `true`, és a dir, és un mètode que aplica al tipus numèric fent un arrodoniment amb el nombre de decimals indicats i transformant el valor numèric en un `string`. Aquest mètode ja existeix, vosaltres no l'heu de crear.

Nota 2: l'àrea d'un triangle és la meitat de la base per alçada. Per calcular l'alçada del `TriangleEquilater` utilitzeu el teorema de Pitàgores

(2 punts)

```
class Punt{
  constructor(x,y){
        this.x=x;
        this.y=y;
  }
}

class TriangleEquilater{
  constructor (posicio, base){
        this.posicio=posicio;
        this.base=base;
  }
  ////////////// Mètodes públics /////////////////
  perimetre(){
        return 3*this.base;
  }
  area(){
        var h= Math.sqrt(3)/2*this.base;
        return this.base*h/2;
  }
  novaPosicio(p){
        this.posicio.x=p.x;
        this.posicio.y=p.y;
        return this;
  }
  zoom(k){
        this.base*=k;
        return this;
  }
  toString(){
        return JSON.stringify(this);
  }
}
```

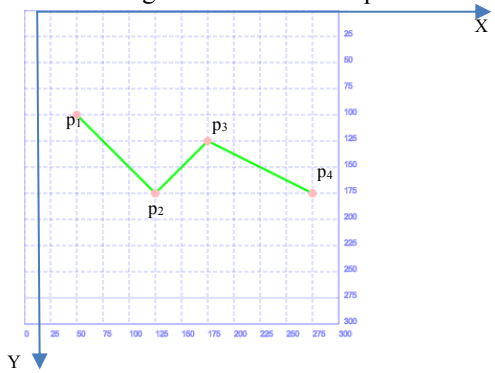
(2 punts) **14 de juliol de 2017.** Amplia la classe TriangleEquilater anterior perquè es pugui dibuixar en un canvas. El dibuix podrà ser amb les cantonades arrodonides amb un radi determinat, vegeu la següent imatge,

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Triangles equilater en el canvas</title>
<style>
canvas{
        display: block;
        margin: 50px auto;
}
</style>
</head>

<body>
<canvas id="canvas" width="500" height="400"></canvas>
<script src="javascript/ampliacioCanvas.js"></script>
<script src="javascript/triangleEquilater.js"></script>

<script>

class TriangleEquilaterCanvas extends TriangleEquilater{
  constructor(posicio,base){
    super/posicio,base);
  }
  draw(ctx, radi){
```

(2 punts)

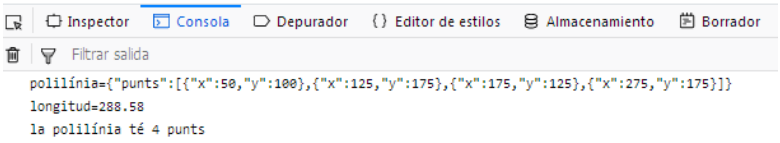
```
    var h=this.base*Math.sqrt(3)/2;
    ctx.save();
    ctx.beginPath();
    ctx.moveTo(this.posicio.x, this.posicio.y);
    ctx.arcTo(this.posicio.x+this.base/2, this.posicio.y  , this.posicio.x           , this.posicio.y-h, radi);
    ctx.arcTo(this.posicio.x            , this.posicio.y-h, this.posicio.x-this.base/2, this.posicio.y  , radi);
    ctx.arcTo(this.posicio.x-this.base/2, this.posicio.y  , this.posicio.x           , this.posicio.y  , radi);
    ctx.lineTo(this.posicio.x,this.posicio.y);
    ctx.stroke();
    ctx.restore();
  }

}

var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

ctx.lineWidth=5;
ctx.strokeStyle="#fbb";
var te=new TriangleEquilaterCanvas(new Punt(250,350),400);
te.draw(ctx,50);

</script>
</body>
</html>
```

(4 punts) Final: **22 de juny de 2018.** Escriviu el codi javascript per a les classes d'objectes `Punt` i `Polilínia`. Una polilínia està caracteritzada per una seqüència de punts que determinen els segments consecutius que conformen la polilínia (vegeu la figura).



Escriviu la classe javascript `Polilinea` que permeti executar el següent codi javascript amb el resultat indicat en la imatge:

```
var pl=new Polilinia();
pl.afegirPunt(new Punt(50,100)).afegirPunt(new Punt(125,175)).afegirPunt(new Punt(175,125)).afegirPunt(new Punt(275,175));
console.log("polilínia="+pl);
console.log("longitud="+pl.longitud().toFixed(2));
console.log("la polilínia té "+pl.quantsPunts()+" punts");
```

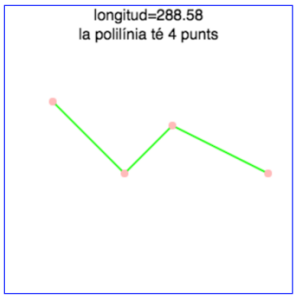


4 punts

```
// En notació ES6

class Punt{
  constructor(x,y){
    this.x=x; this.y=y;
  }
  static distancia(p1,p2){
    return Math.sqrt(Math.pow(p2.x-p1.x,2) + Math.pow(p2.y-p1.y,2));
  }
}


class Polilinia{
  constructor(){
    this.punts=[];
  }
  quantsPunts(){
    return this.punts.length;
  }
  afegirPunt(p){
    this.punts.push(p);
    return this;
  }
  longitud(){
    var l=0;
    for(var i=0; i<this.punts.length-1; i++){
        l+=Punt.distancia(this.punts[i],this.punts[i+1]);
    }
    return l;
  }
  toString(){
    return JSON.stringify(this);
  }
}
```

(3 punts) **22 de juny de 2018.** Escriviu una "ampliació" de la Polilínia de la pregunta anterior perquè es dibuixi en un canvas. Observeu el resultat que s'obté després d'executar el codi que s'indica:

- Els punts es dibuixen amb circumferències de 4 píxels de radi i color #fbb
- La font del text és de 16 píxels, sans-serif: la primera línia a la coordenada y=10 i la segona línia a la coordenada y=30
- El segment es dibuixa amb un gruix de 2 píxels i de color #bfb




```
<!DOCTYPE HTML>
<html>
<head><meta charset="utf-8"><title>Polilínia</title>
<style>
canvas{  display:block;
         border:thin #00F solid;
         margin: 0 auto;
}
</style>
</head>
<body>
<canvas id="canvas" width="300" height="300"></canvas>
<script src="javascript/polilinia.js"></script> <!-- codi de la pregunta anterior -->
<script>
class PuntCanvas extends Punt {
```

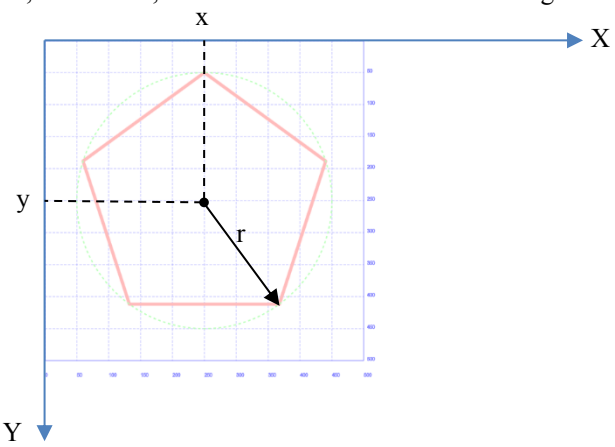
(1 punt)

```
  constructor(x, y, ctx) {
    super(x, y);
    this.ctx = ctx;
  }
  draw() {
    this.ctx.save();
    this.ctx.beginPath();
    this.ctx.fillStyle = "#fbb";
    this.ctx.arc(this.x, this.y, 4, 0, 2 * Math.PI);
    this.ctx.fill();
    this.ctx.restore();
  }
}

class PoliliniaCanvas extends Polilinia{
  constructor(ctx){
    super();
    this.ctx=ctx;
  }
  draw(){
```

(2 punts)

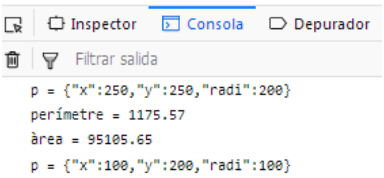
```
       this.ctx.save();
       this.ctx.lineWidth=2;
       this.ctx.strokeStyle="#0f0";
       this.ctx.beginPath();
      // els segments
       this.ctx.moveTo(this.punts[0].x,this.punts[0].y);
       for(var i=0; i<this.punts.length; i++){
         this.ctx.lineTo(this.punts[i].x,this.punts[i].y);
       }
      // els punts
       this.ctx.stroke();
       for(var i=0; i<this.punts.length; i++){
         this.punts[i].draw();
       }
       // el text
       this.ctx.font='16px sans-serif';
       this.ctx.textAlign="center";
       this.ctx.textBaseline="middle";
       this.ctx.fillText("longitud="+this.longitud().toFixed(2), this.ctx.canvas.width/2, 10);
       this.ctx.fillText("la polilínia té "+this.quantsPunts()+" punts", this.ctx.canvas.width/2, 30);
       this.ctx.restore();
  }
}
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var pl=new PoliliniaCanvas(ctx);
pl.afegirPunt(new Punt(50,100,ctx)).afegirPunt(new Punt(125,175,ctx)).afegirPunt(new Punt(175,125,ctx)).afegirPunt(new Punt(275,175,ctx));
pl.draw();

</script>
</body>
</html>
```

(4 punts) **Recuperació: 11 de juliol de 2018.** Escriviu el codi javascript per a la classe d'objectes `Pentagon`. Un `Pentagon` és un polígon regular de 5 costats. El `Pentagon` el podem considerar inscrit dins una circumferència. Així, el `Pentagon` tindrà una posició, la del centre de la circumferència que el conté, i una mida, la del radi de la circumferència. Vegeu la imatge següent:



Escriviu la classe javascript `Pentagon` que permeti executar el següent codi javascript amb el resultat indicat en la imatge:

```
var p=new Pentagon(250,350,400);
console.log("p = "+p);
console.log("perímetre = "+p.perimetre().toFixed(2));
console.log("àrea = "+p.area().toFixed(2));
p.novaPosicio(100,200).zoom(1/2);  // hem canviat la posició i el reduim a la meitat
console.log("p = "+p);
```



Nota 1: `(3.45678).toFixed(2)== "3.46"` és `true`, és a dir, és un mètode que aplica al tipus numèric fent un arrodoniment amb el nombre de decimals indicats i transformant el valor numèric en un `string`. Aquest mètode ja existeix, vosaltres no l'heu de crear.

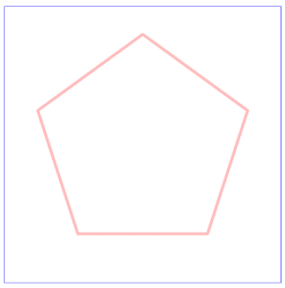
Nota 2: l'àrea d'un pentàgon és $A = \frac{5}{2}r^2 \sin(72^o)$ i el perímetre és $P = 10r^2 \sin(36^o)$

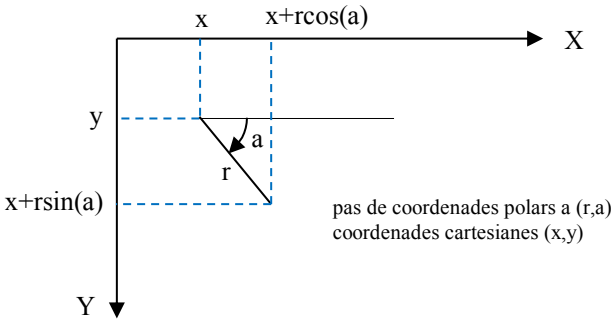
(4 punts)

```
// Solució en ES6

class Pentagon{
   constructor(x,y,radi){
      this.x=x;
      this.y=y;
      this.radi=radi;
   }
////////////// Mètodes públics /////////////////
   perimetre(){
            return 5*2*this.radi*Math.cos(Math.PI/180*54);  // recordeu que sin(a)=cos(90-a)
   }
   area(){
            return 5/2*Math.pow(this.radi,2)*Math.sin(Math.PI/180*72); // sin(2a)=2sin(a)cos(a)
   }
   novaPosicio function(x,y){
            this.x=x;
            this.y=y;
            return this;
   }
   zoom(k){
            this.radi*=k;
            return this;
   }
   toString(){
            return JSON.stringify(this);
   }
}
```

(3 punts) **Recuperació: 11 de juliol de 2018.** Amplia la classe `Pentagon` anterior perquè es pugui dibuixar en un canvas. Observeu l'imatge del resultat d'executar el programa i l'imatge de com passar de coordenades polars a coordenades cartesianes,

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Pentàgon</title>
<style>
canvas{
  border: 1px solid #00F;
  display: block;
  margin: 50px auto;
}
</style>
</head>

<body>
<canvas id="canvas" width="500" height="500"></canvas>
<script type="text/javascript" src="javascript/pentagon.js"></script> <!-- codi de la pregunta anterior-->

<script type="application/javascript">

class PentagonCanvas extends Pentagon{
  constructor(x,y,radi,ctx){
    super(x,y,radi);
    this.ctx=ctx;
  }
  draw(){
```





(3 punts)

```
  var dA=2*Math.PI/5;    // angle de la rotació
  this.ctx.save()
  this.ctx.beginPath();
// mètode 1, utilitzan la simetria circular del pentàgon
/*
  this.ctx.translate(this.x,this.y); // posem l'origen de coordenades al centre del cercle que ha de contenir el poligon
  this.ctx.rotate(-Math.PI/2);
  this.ctx.moveTo(this.radi,0);
  for(var i=0; i<5; i++){
        this.ctx.rotate(dA);
        this.ctx.lineTo(this.radi,0);
  }
*/
// mètode 2, calculant els punts vèrtex del pentàgon
  var punts=[];
  var angle=-Math.PI/2;
  // calculem tots els vèrtexs del poligon
  for(var i=0; i<5; i++){
        punts.push({x:this.x+this.radi*Math.cos(angle), y:this.y+this.radi*Math.sin(angle)});
        angle+=dA;
  }
  // tracem les línies
  this.ctx.moveTo(punts[0].x,punts[0].y);
  for(i=0; i<5; i++){
        this.ctx.lineTo(punts[i].x,punts[i].y);
  }

  this.ctx.lineJoin="round";
  this.ctx.closePath();
  this.ctx.stroke();
  this.ctx.restore();

  }
}
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

ctx.lineWidth=5;
ctx.strokeStyle="#fbb";
var p=new PentagonCanvas(250,250,200);
p.draw();

</script>
</body>
</html>
```

(4 punts) **Final: 25 de juny de 2019.** Escriviu el codi javascript per a la "classe" d'objectes `Polinomi`. Un polinomi, *p(x)*, és una funció suma de potències naturals de x i ve determinat pels seus coeficients a_i amb $i=0..n$,

$$p(\mathrm{x}) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n$$

Donats dos polinomis, $p_1(x)$ i $p_2(x)$,

$$p_1(\mathrm{x}) = \sum_{i=0}^{n} a_i x^i , \qquad p_2(\mathrm{x}) = \sum_{i=0}^{m} b_i x^i$$

es poden sumar,

$$p_1(\mathrm{x}) + p_2(\mathrm{x}) = \sum_{i=0}^{m} (a_i + b_i) x^i \quad per\ a \quad m \geq n$$

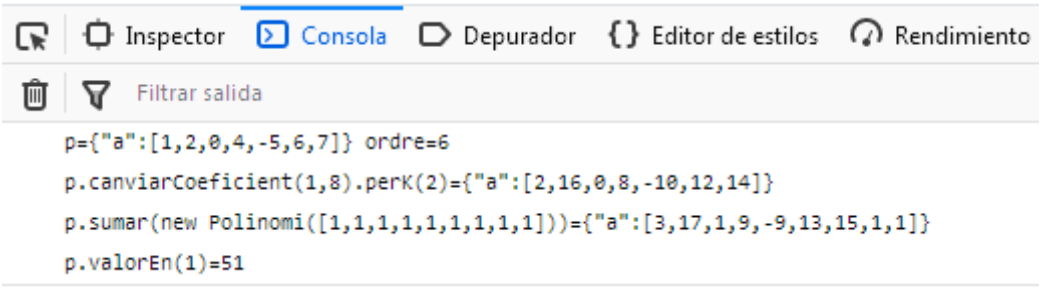
Un polinomi es pot multiplicar per una constant,

$$k\ p(\mathrm{x}) = \sum_{i=0}^{n} k a_i x^i$$

i, es pot determinar el valor del polinomi en un valor concret de *x*,

Escriviu la classe javascript `Polinomi` que permeti executar el següent codi javascript amb el resultat indicat en la imatge:

```
var p=new Polinomi([1,2,0,4,-5,6,7]);

console.log("p="+p+" ordre="+p.ordre());
console.log("p.canviarCoeficient(1,8).perK(2)="+p.canviarCoeficient(1,8).perK(2));
console.log("p.sumar(new Polinomi([1,1,1,1,1,1,1,1]))="+p.sumar(new Polinomi([1,1,1,1,1,1,1,1])));
console.log("p.valorEn(1)="+p.valorEn(1));
```



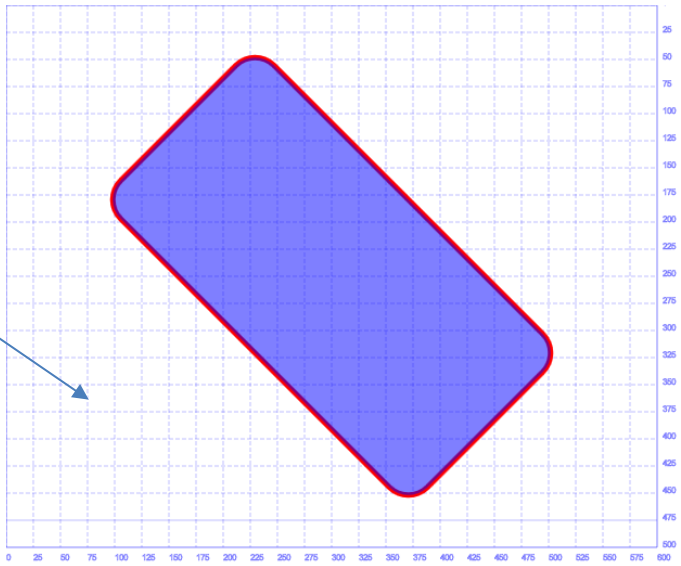
4 punts

```
// ES6
class Polinomi{
  constructor(coeficients){
    this.a=coeficients; // un array
  }

  sumar(p2){
    // segona solució: recorregut del polinomi p2
    for(let i=0; i<=p2.ordre(); i++){
      if(i<=this.ordre()){
       this.a[i]+=p2.a[i];
      }
      else{
       this.a[i]=p2.a[i];
      }
    }
    return this;
  }
  ordre(){return this.a.length-1;}
  canviarCoeficient(index, coeficient){this.a[index]=coeficient; return this;}
  perK(k){
    for(let i in this.a){
      this.a[i] *=k;
    }
    return this;
  }
  valorEn(x){
    var r=0;
    for(let i in this.a){
      r +=this.a[i]*Math.pow(x,i);
    }
    return r;
  }

  toString(){
    return JSON.stringify(this);

  }
}
```

(3 punts) **Final: 25 de juny de 2019.** Escriviu el mètode plot de la "classe" `Rectangle` següent, per poder obtenir el dibuix del rectangle dins un `canvas`. Vegeu l'exemple d'execució següent:

```
:
<canvas id="canvas" width="600" height="500"></canvas>
<script>
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");

  var r=new Rectangle(300,250,400,200,25,45);
  r.plot(ctx,"#f00",5,"rgba(0,0,255,0.5)");
</script>
:
```

el mètode *plot* no ha de dibuixar la graella, la mostrem només a títol informatiu

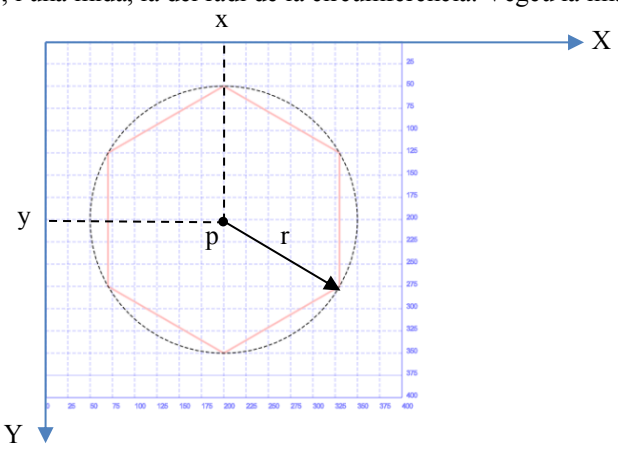


```
class Rectangle{
  constructor(x,y,amplada,altura,radi,angle){
    this.x=x; this.y=y;            // posició, centre del rectangle
    this.w=amplada; this.h=altura; // mides
    this.r=radi;                   // radi d'arrodinament cantonades
    this.angle=angle;              // angle de rotació en graus. Angle positiu, sentit horari
  }

  plot(ctx,colorVora,gruixVora,colorDins){
```

(3 punts)

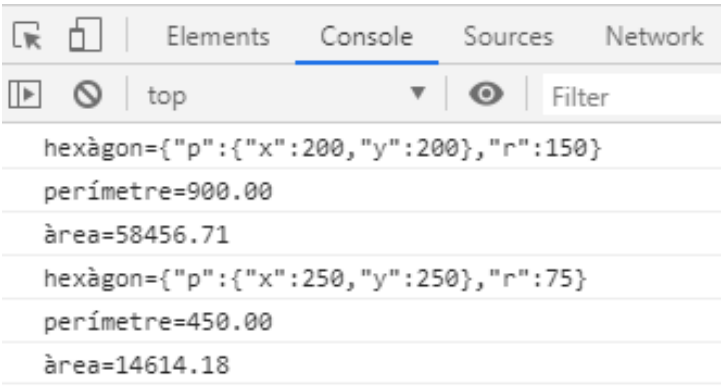
```
    ctx.save();
    ctx.strokeStyle=colorVora;
    ctx.lineWidth=gruixVora;
    ctx.fillStyle=colorDins;
    ctx.beginPath();
    ctx.translate(this.x,this.y);
    ctx.rotate(Math.PI/180*this.angle);
    var p=[{x:-this.w/2, y: this.h/2},
           {x: this.w/2, y: this.h/2},
           {x: this.w/2, y:-this.h/2},
           {x:-this.w/2, y:-this.h/2}
          ]
    ctx.moveTo(p[0].x+this.r,p[0].y);
    ctx.arcTo(p[1].x, p[1].y, p[2].x, p[2].y, this.r);
    ctx.arcTo(p[2].x, p[2].y, p[3].x, p[3].y, this.r);
    ctx.arcTo(p[3].x, p[3].y, p[0].x, p[0].y, this.r);
    ctx.arcTo(p[0].x, p[0].y, p[1].x, p[1].y, this.r);
    ctx.stroke();
    ctx.closePath();
    ctx.fill();
    ctx.restore();


  }
}
```

(4 punts) **Recuperació: 10 de juliol de 2019.** Escriviu el codi javascript per a la "classe" d'objectes `Hexagon`. Un `Hexagon` és un polígon regular de 6 costats. El `hexagon` el podem considerar inscrit dins una circumferència. Així, el `Hexagon` tindrà una posició, la del centre de la circumferència que el conté, i una mida, la del radi de la circumferència. Vegeu la imatge següent:



Escriviu la classe javascript `Hexagon` que permeti executar el següent codi javascript amb el resultat indicat en la imatge:

```
var hexagon=new Hexagon(new Punt(200,200),150);

console.log("hexàgon="+hexagon);
console.log("perímetre="+hexagon.perimetre().toFixed(2));
console.log("àrea="+hexagon.area().toFixed(2));
console.log("hexàgon="+hexagon.canviarPosicio(new Punt(250,250)).zoom(1/2));
console.log("perímetre="+hexagon.perimetre().toFixed(2));
console.log("àrea="+hexagon.area().toFixed(2));
```



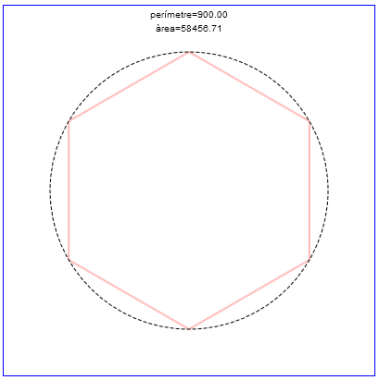
Nota 1: `(3.45678).toFixed(2)== "3.46"` és `true`, és a dir, és un mètode que aplica al tipus numèric fent un arrodoniment amb el nombre de decimals indicats i transformant el valor numèric en un `string`. Aquest mètode ja existeix, vosaltres no l'heu de crear.

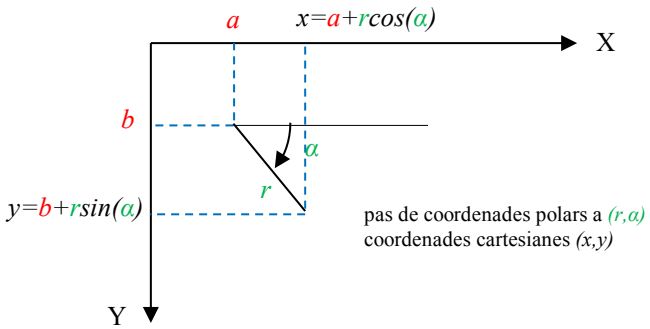
Nota 2: l'àrea d'un hexàgon és $A = 3r^2 \sin(60^o)$ i el perímetre és $P = 12r \sin(30^o)$

(4 punts)

```
// En notació ES6
class Punt{
  constructor(x,y){
    this.x=x; this.y=y;
  }
}

class Hexagon{
  constructor(posicio,mida){
    this.p=posicio;   // posició, el centre del cercle que conté l'hexàgon
    this.r=mida;      // mida, donada pel radi del cercle que conté l'hexàgon
  }
  perimetre(){return 12*this.r*Math.sin(Math.PI/6);}
  area(){return 3*Math.pow(this.r,2)*Math.sin(2*Math.PI/6);}
  canviarPosicio(posicio){
    this.p=posicio;
    return this;
  }
  zoom(k){
    this.r*=k;
    return this;
  }
  toString(){return JSON.stringify(this);}

}


// En notació ES3
function Punt(x,y){
        this.x=x;
        this.y=y;
}


function Hexagon(posicio,mida){
  this.p=posicio;   // posició, el centre del cercle que conté l'hexàgon
  this.r=mida;      // mida, donada pel radi del cercle que conté l'hexàgon
}

////////////// Mètodes públics /////////////////
Hexagon.prototype={
  constructor: Hexagon,

  perimetre: function(){
    return 12*this.r*Math.sin(Math.PI/6);
  },
  area: function(){
    return 3*Math.pow(this.r,2)*Math.sin(2*Math.PI/6);
  },
  canviarPosicio: function(posicio){
    this.p=posicio;
    return this;
  },
  zoom: function(k){
    this.r*=k;
    return this;
  },
  toString: function(){
    return JSON.stringify(this);
  }
};
```

(3 punts) **Recuperació: 10 de juliol de 2019.** Amplia la "classe" `Hexagon` anterior perquè es pugui dibuixar en un canvas. Observeu la imatge del resultat d'executar el programa i la imatge de com passar de coordenades polars a coordenades cartesianes, per si us cal calcular els punts:



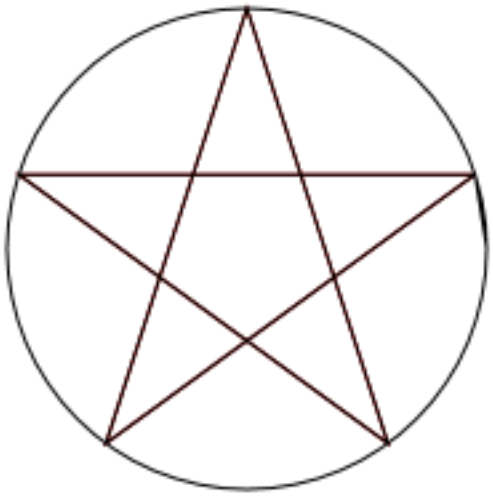



```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Pentàgon</title>
<style>
canvas{
  border: 1px solid #00F;
  display: block;
  margin: 50px auto;
}
</style>
</head>

<body>
<canvas id="canvas" width="400" height="400"></canvas>
<script type="text/javascript" src="javascript/hexagon.js"></script> <!-- codi de la pregunta anterior-->

<script type="application/javascript">
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var hexagon=new Hexagon(new Punt(200,200),150);
hexagon.draw(ctx);
Hexagon.prototype.draw=function(ctx){
```

(3 punts)

```
  var dA=2*Math.PI/6;    // angle de la rotació
  ctx.save();
  ctx.lineWidth=2;
  ctx.strokeStyle="#fbb";
  ctx.beginPath();
 // mètode 1, utilitzan la simetria circular del pentàgon
  ctx.translate(this.p.x,this.p.y); // posem l'origen de coordenades al centre del cercle que conté l'hexàgon
  ctx.rotate(-Math.PI/2);
  ctx.moveTo(this.r,0);
  for(let i=0; i<6; i++){
    ctx.rotate(dA);
    ctx.lineTo(this.r,0);
  }
/* // mètode 2, calculant els punts vèrtex de l'hexàgon
  var punts=[];
  var angle=-Math.PI/2;
  // calculem tots els vèrtexs de l'hexàgon
  for(let i=0; i<6; i++){
    punts.push(new Punt(this.p.x+this.r*Math.cos(angle), this.p.y+this.r*Math.sin(angle)));
    angle+=dA;
  }
  // tracem les línies
  ctx.moveTo(punts[0].x,punts[0].y);
  for(let i=0; i<6; i++){
    ctx.lineTo(punts[i].x,punts[i].y);
  }
*/
  ctx.lineJoin="round";
  ctx.closePath();
  ctx.stroke();
  ctx.restore();
  ctx.save();
  ctx.font='10px sans-serif';   // el text
  ctx.textAlign="center";
  ctx.textBaseline="middle";
  ctx.fillText("perímetre="+this.perimetre().toFixed(2), ctx.canvas.width/2, 10);
  ctx.fillText("àrea="+this.area().toFixed(2), ctx.canvas.width/2, 25);
  ctx.restore();
  ctx.save();
  ctx.setLineDash([4,2]);   // el cercle
  ctx.beginPath();
  ctx.arc(this.p.x,this.p.y,this.r,0,2*Math.PI);
  ctx.stroke();
  ctx.restore();

}
</script>
</body></html>
```

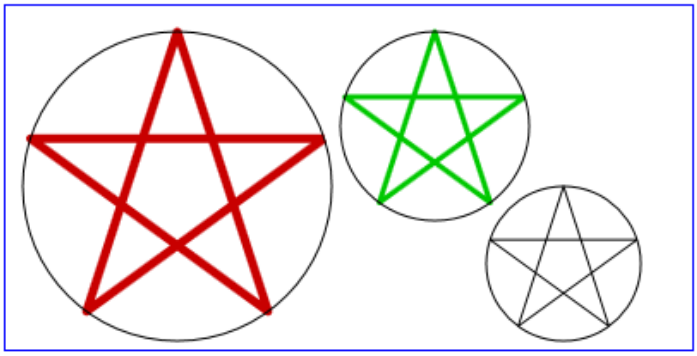
(5 punts) **Recuperació: 10 de juliol de 2015.** Escriviu una funció javascript que permeti dibuixar pentacles en un canvas. Un pentacle és un estel de 5 puntes que s'inscriu en un pentàgon regular. Observeu el resultat que s'obté amb tres crides successives a aquesta funció.



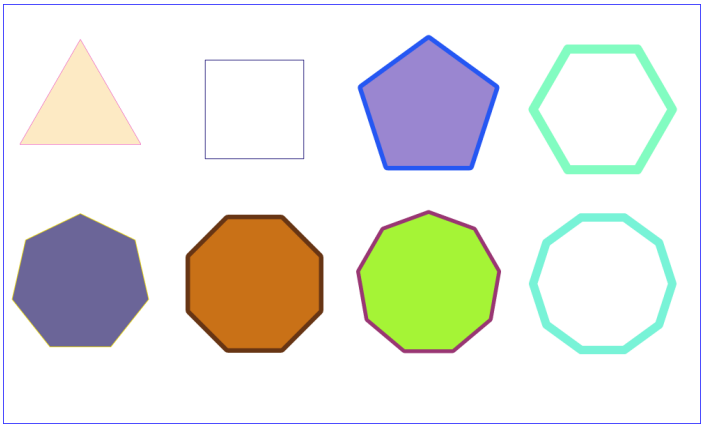
```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Pregunta 4</title>
<style>
canvas{border:thin solid #00F;}
</style>
</head>

<body>
<canvas id="canvas" width="400" height="200"></canvas>
<script type="application/javascript">

var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

function pentacle (ctx, x, y,radi, colorLinia, gruixLinia){

    // primer cal calcular els angles. Atès que hi ha 5 puntes, el increment d'angle és 360/5=72
    // si volem que la segona punta de l'estel sigui la que mira al nord (recorregut en sentit antihorari),
    // l'angle de la primera punta serà 90-72=18.

    // recordeu com es passa de coordenades polars (r,α) a cartesianes (x,y):
    // x= rcos(α), y=rsin(α)
```

(5 punts)

```
  var angles=[18,90,162,234,306];
  var coordenades=[];
  for(var i=0; i<angles.length; i++){
      coordenades[i]={x:radi*Math.cos(angles[i]*Math.PI/180), y:-radi*Math.sin(angles[i]*Math.PI/180)};
  }
  ctx.save();
  ctx.lineJoin="round";
  ctx.beginPath();
  ctx.moveTo(x+coordenades[0].x,y+coordenades[0].y);
  for(var i=0, k=2; i<angles.length; i++, k=(k+2)%5){
      ctx.lineTo(x+coordenades[k].x,y+coordenades[k].y);
  }
  if(colorLinia) ctx.strokeStyle=colorLinia;
  if(gruixLinia) ctx.lineWidth=gruixLinia;
  ctx.stroke();
  ctx.restore();
  ctx.beginPath();
  ctx.arc(x, y, radi, 0, 2*Math.PI);
  ctx.stroke();

// mètode dos
//  ctx.save();
//  ctx.lineJoin="round";
//  ctx.translate(x,y);
//  ctx.rotate(-18*Math.PI/180);
//  ctx.beginPath();
//  ctx.moveTo(radi,0);
//  for(var i=0; i<5; i++){
//    ctx.rotate(2*72*Math.PI/180);
//    ctx.lineTo(radi,0);
//  }
//  ctx.save();
//  if(colorLinia) ctx.strokeStyle=colorLinia;
//  if(gruixLinia) ctx.lineWidth=gruixLinia;
//  ctx.stroke();
//  ctx.restore();
//  ctx.beginPath(); ctx.arc(0, 0, radi, 0, 2*Math.PI); ctx.stroke();
//  ctx.restore();

}
  pentacle(ctx, 100, 105, 90, "rgb(200,0,0)", 5);
  pentacle(ctx, 250, 70, 55, "rgb(0,200,0)", 3);
  pentacle(ctx, 325, 150, 45 );</script>

</body>
</html>
```



(2.5 punts) **Final: 29 de juny de 2016**. Escriviu una funció javascript que permeti dibuixar en un canvas polígons regulars de n costats, inscrits dins un cercle de radi donat el centre del qual està a la posició (x,y). Observeu el resultat que s'obté després d'executar el codi que s'indica:

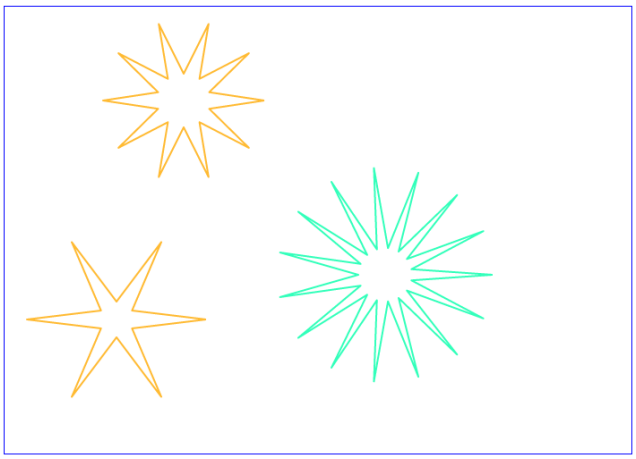


```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Pregunta 4</title>
<style>
canvas{border:thin solid #00F;}
</style>
</head>

<body>
<canvas id="canvas" width="1000" height="600"></canvas>
<script type="application/javascript">

  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  for(var i=0; i<4; i++){
    for(var j=0; j<2; j++){
        ctx.strokeStyle="rgb(" + atzar(255) + "," + atzar(255) + "," + atzar(255) + ")";
        ctx.lineWidth= atzar(15);
        ctx.fillStyle="rgba(" + atzar(255) + "," + atzar(255) + "," + atzar(255) + "," + atzar(1) + ")";

        poligonRegular(ctx, 3+i+4*j,  110+i*250, 150+j*250, 100);
    }
  }

  function poligonRegular(ctx, n, x, y, radi){
```

(2,5 punts)

```
    var metode1=false;
    var dA=2*Math.PI/n;    // angle de la rotació
    ctx.save()
    ctx.beginPath();
    if(metode1){
       ctx.translate(x,y);  // posem l'origen de coordenades al centre del cercle que ha de contenir el poligon
       ctx.rotate(-Math.PI/2);
       if(n%2==0) ctx.rotate(dA/2);
       ctx.moveTo(radi,0);
       for(var i=0; i<n; i++){
         ctx.rotate(dA);
         ctx.lineTo(radi,0);
       }
    }
    else{  // mètode 2, calculant les coordenades dels vèrtexs del poligon
       var punts=[];
       var angle=-Math.PI/2;
       if(n%2==0) angle+=dA/2;
       // calculem tots els vèrtexs del poligon
       for(var i=0; i<n; i++){
         punts.push({x:x+radi*Math.cos(angle), y:y+radi*Math.sin(angle)});
         angle+=dA;
       }
       // tracem les línies
       ctx.moveTo(punts[0].x,punts[0].y);
       for(i=0; i<n; i++){
         ctx.lineTo(punts[i].x,punts[i].y);
       }
    }
    ctx.lineJoin="round";
    ctx.closePath();
    ctx.stroke();
    ctx.fill();
    ctx.restore();
  }

  function atzar (b){return Math.floor(Math.random()*(b+1));}

</script>
</body>
</html>
```

(2.5 punts) **Recuperació: 13 de juliol de 2016**. Escriviu una funció javascript que permeti dibuixar en un canvas estels de n puntes, inscrits dins un cercle de radi donat el centre del qual està a la posició (x,y). Les puntes de l'estel van des d'un cercle de radi1 a un cercle de radi2 on radi1<radi2. Observeu el resultat que s'obté després d'executar el codi que s'indica:



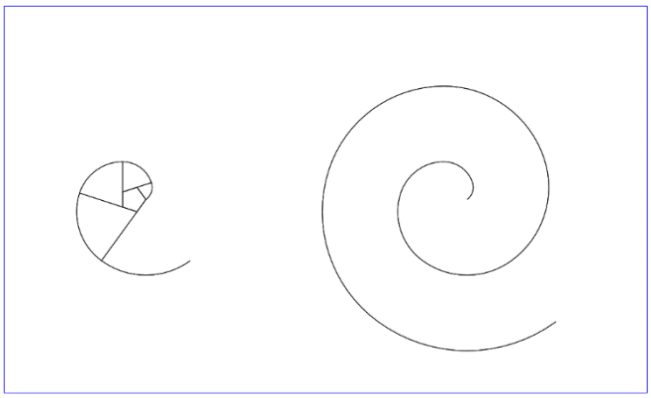
```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Pregunta 4</title>
<style>
canvas{border:thin solid #00F;}
</style>
</head>

<body>
<canvas id="canvas" width="700" height="500"></canvas>
<script type="application/javascript">

  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");


  function estel(ctx, n, x, y,radi1, radi2){
```

(2,5 punts)

```
      var angle=Math.PI/n;
      ctx.save()
      ctx.lineJoin="round";
      ctx.translate(x,y);
     ctx.beginPath();
     ctx.moveTo(radi2,0);
     for(var i=0; i<2*n; i++){
       ctx.rotate(angle);
       if(i%2==0)
         ctx.lineTo(radi1,0);
       else
         ctx.lineTo(radi2,0);
     }
     ctx.stroke();
     ctx.restore();
```

```
  }

  ctx.strokeStyle="#fb3";
  ctx.lineWidth=2;
  estel(ctx, 10, 200, 105, 30, 90);
  estel(ctx, 6, 125, 350, 20, 100);
  ctx.strokeStyle="#3fb";
  estel(ctx, 15, 425, 300, 30, 120 );

</script>
</body>
</html>
```

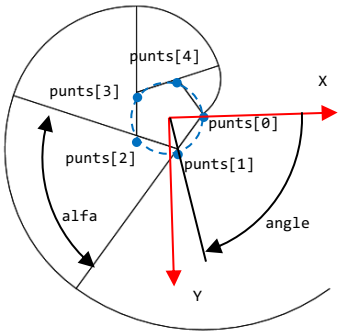
(3 punts) Recuperació: **14 de juliol de 2017.** Escriviu una funció javascript que permeti dibuixar en un canvas espirals de n voltes, a partir d'un polígon regular inscrit en un cercle de radi donat el centre del qual està a la posició (x,y). Observeu el resultat que s'obté després d'executar el codi que s'indica:



```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Espirals en el canvas</title>
<style>
canvas{
        border:thin solid #00F;
        display:block;
        margin: 50px auto;
}
</style>
</head>

<body>
<canvas id="canvas" width="1000" height="600"></canvas>
<script type="application/javascript">

var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

espiral(ctx, 5,  200, 300, 20, true, 1); // n==5 => pentàgon inscrit en un cercle de radi 20 i centre en (200,300)
                                         // true => es veu el pentàgon i els costats allargats del mateix.
                                         // 1 => només una volta
espiral(ctx, 5,  700, 300, 20, false, 2);

function espiral(ctx, n, x, y, radi, veurePoligon, nVoltes){
        var alfa=2*Math.PI/n;    // angle del sector d'arc
        var costat;              // mida d'un costar del polígon regular
        ctx.save()
        var punts=[];
        var angle=0;
       // calculem els punts del polígon i els guardem a l'array punts
       // utilitzeu coordenades polars: x=r*cos(angle), y=r*sin(angle)
```

(1 punt)

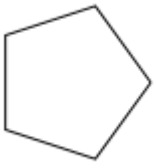
```
    for(var i=0; i<n; i++){
            punts.push({x:x+radi*Math.cos(angle), y:y+radi*Math.sin(angle)});
            angle+=alfa;
    }
```




```
     costat=distancia(punts[0], punts[1]); // calculem la mida d'una aresta del polígon
     // tracem el polígon
    if(veurePoligon){
```

(1 punt)

```
            ctx.beginPath();
            ctx.moveTo(punts[0].x,punts[0].y);
            for(i=0; i<n; i++){
                    ctx.lineTo(punts[i].x,punts[i].y);
            }
            ctx.lineTo(punts[0].x,punts[0].y);
            ctx.stroke();
```

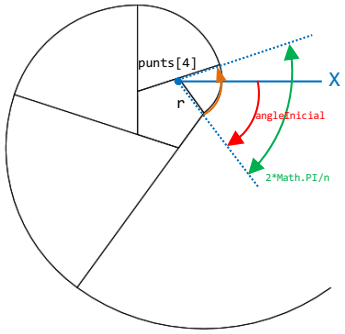


```
    }
```

```
        var angleInicial=Math.atan2(Math.abs(punts[0].y-punts[punts.length-1].y),
                                    Math.abs(punts[0].x-punts[punts.length-1].x));
        var angleFinal;
        var r=costat;

       // dibuixem les nVoltes d'espiral
        for(var v=0; v<nVoltes; v++){
         // cal dibuixar un arc des de cadascun dels punts (en sentit antihorari) del poligon.
         // En el dibuix teniu el primer sector, punts[punts.length-1], dibuixat
```

(1 punt)

```
          for(var i=punts.length-1; i>=0; i--){ // sentit antihorari
                angleFinal=angleInicial-2*Math.PI/n;
                if(veurePoligon){
                  ctx.beginPath();
                  ctx.moveTo(punts[i].x,punts[i].y);  // ens posem en el punt
                }
                else{
                  ctx.moveTo(punts[i].x,punts[i].y);
                  ctx.beginPath();
                }
                ctx.arc(punts[i].x,punts[i].y, r, angleInicial, angleFinal, true);
                ctx.stroke();
                angleInicial=angleFinal;
                r+=costat;
          }
```




```
        }
        ctx.lineJoin="round";
        ctx.restore();
}

function distancia(p1,p2){
        return Math.sqrt(Math.pow(p2.y-p1.y,2)+Math.pow(p2.x-p1.x,2));
}

</script>
</body>
</html>
```