Considere a definição dos tipos abaixo para representar um ponto no espaço 2D e um círculo:

```
struct ponto {
  float x;
  float y;
};
typedef struct ponto Ponto;

struct circ {
  float r;   /* raio */
  Ponto p;   /* centro */
};
typedef struct circ Circ;
```

Para todas as funções pedidas abaixo, não se esqueça de acrescentar um teste numa função `main`.

(a) Implemente uma função que calcula a distância entre dois pontos, seguindo o protótipo:

```
float distancia (Ponto* a, Ponto* b);
```

(b) Implemente uma função para calcular a área de um círculo, seguindo o protótipo:

```
float circ_area (Circ* c);
```

(c) Usando a função do item (a), implemente uma função que recebe um círculo e um ponto e retorna se o ponto está dentro do círculo, seguindo o protótipo:

```
int circ_pertence (Circ* c, Ponto* p);
```

(d) Usando a função do item (a), implemente uma função que verifica se dois círculo se interceptam. A função deve retorna 1 se há interseção e o caso contrário, seguindo o protótipo:

```
int circ_intersecao (Circ* a, Circ* b);
```

(e) Defina um tipo para representar *retângulos* (`Rect`) definidos por dois pontos. Implemente as mesmas funções dos itens (b), (c) e (d) aplicados para este tipo retângulo, seguindo os protótipos:

```
float rect_area (Rect* r);
int rect_pertence (Rect* r, Ponto* p);
int rect_intersecao (Rect* a, Rect* b);
```

(f) Implemente uma função que recebe um retângulo como parâmetro e retorne o maior círculo circunscrito neste retângulo, seguindo o protótipo:

```
Circ circunscrito (Rect* r);
```

(g) Altere a função do item anterior para a função alocar dinamicamente o círculo a ser retornado, seguindo o protótipo:

```
Circ* cria_circunscrito (Rect* r);
```