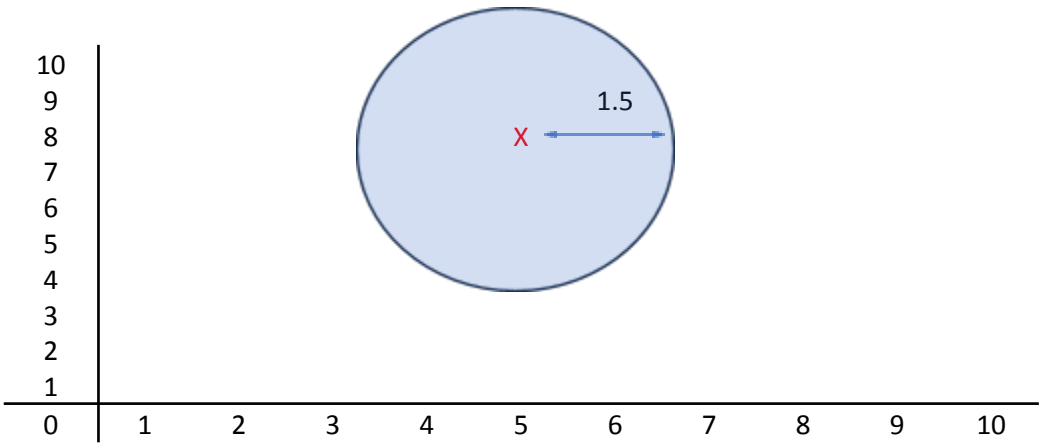
Uma equipe de desenvolvimento de jogos deseja selecionar um grupo de usuários para os testes iniciais de um novo jogo.

Para isso elaborou um questionário onde o usuário informa sua idade e atribui uma nota entre 0 e 10 a dois jogos já existentes no mercado. Considerando que o testador ideal é o usuário que deu nota 5 ao primeiro jogo e nota 8 ao segundo jogo, mostre todos os candidatos que estão próximos do testador ideal, isto é, cuja distância é de até 1,5 do testador ideal



TAD CANDIDATO

a) Usando o TAD Ponto para representar as notas atribuídas aos jogos, defina o TAD Candidato
b) O tipo Candidato tem: nome, idade e um ponto representado as notas que deu ao jogo
c) Ao criar um candidato tem-se seu nome, idade e nota dada aos dois jogos.
d) Funcionalidades desejadas:

- Criar, liberar,exibir, retornar string
- get para cada atributo (desejável)
- set para cada atributo (desejável)
- ehTestador: recebe a pontuação para cada jogo do testador ideal e um limiar. Retorna True se o candidato pode ser um testador do novo jogo ou 0, caso contrário. (nesse exemplo: nota 5 ao primeiro jogo e nota 8 ao 2º) Para ser testador, as notas do candidato devem estar a uma distância menor ou igual ao limiar das notas do testador ideal.
- Compara : recebe dois candidatos e retorna -1, se o primeiro é mais jovem que o segundo, 0 se tem a mesma idade e +1,caso contrário

MAIN

Construa um programa que - monte e preencha um vetor de ponteiros para n (perguntado ao usuário) candidatos.

Obs: Utilize a função sorteio abaixo para simular a idade, nota ao jogo 1, nota ao jogo 2 eo índice do vetor de nomes

char vNome[10][5]={"Dodo","Dede","Didi","Zizo","Vava","Dudu","Mimi","Lala","Lele"};
// Sorteia um número inteiro entre lInf e lSup inclusive

```
int sorteio(int lInf,int lSup){
  return (rand()%((lSup-lInf)+1))+lInf;
```

Construa um novo vetor com ponteiros para os candidatos que podem ser testadores

Mostre os selecionados, ordenados crescentemente por idade.

TADPonto.h

```
/* Tipo exportado */
typedef struct ponto Ponto;

/* Funções exportadas */
/* Função cria - Aloca e retorna um ponto com coordenadas
(x,y) */
Ponto* pto_cria(float x, float y);

/* Função libera - Libera a memória de um ponto previamente
criado */
void pto_libera(Ponto* p);

/* Função acessa - Retorna os valores das coordenadas de um
ponto */
void pto_acessa(Ponto* p, float* x, float* y);

/* Função atribui - Atribui novos valores às coordenadas de
um ponto */
void pto_atribui(Ponto* p, float x, float y);

/* Função distancia - Retorna a distância entre dois pontos
*/
float pto_distancia(Ponto* p1, Ponto* p2);

/* Função getPonto - Retorna uma string com os valores do
ponto formato(x,y)*/
char* pto_getPonto(Ponto* p);

/* Função exibe -Escreve na tela o ponto no formato
(xxx.xx,yyy.yy)*/

/* Função exibe -Escreve na tela o ponto no formato
(xxx.xx,yyy.yy)*/
void pto_exibe(Ponto* p);

int pto_compara(Ponto* pt1,Ponto* pt2);
/* Função compara -recebe dois pontos e retorna:
0 se os pontos são iguais
```

```
valor negativo se o ponto 1 1 está mais próximo da origem que
o pto 2
valor positivo,  cc*/


/* Função pto_soma, recebe dois pontos e retorna um
novo ponto, com as soma das coordenadas */
Ponto* pto_soma(Ponto * p1, Ponto* p2);
```