Escreva um programa em C no Linux usando a biblioteca Pthreads e que atenda aos seguintes requisitos:

R1. O programa recebe três parâmetros $m$, $n$ e $r$ na linha de comando; ou seja, o programa deve ser invocado como

```
$ ./prog m n r
```

R2. O programa deve alocar dinamicamente uma matriz $A$ de $m \times n$ números inteiros, onde $m$ é o número de linhas e $n$ é o número de colunas, e preenchê-la com números entre 1 e $m \times n$. Os números devem estar em posições aleatórias (isto é, dispostos de forma não sequencial), e não podem ser repetidos (sugestão: preencha a matriz sequencialmente e depois faça um rearranjo aleatório dos elementos). Por exemplo, para $m = 2$ e $n = 3$, a matriz abaixo seria válida:

$$\begin{pmatrix} 5 & 2 & 1 \\ 6 & 3 & 4 \end{pmatrix}$$

R3. O programa deve criar quatro *threads*, todas elas executando a mesma função (descrita mais à frente).

R4. Cada *thread* deve receber como parâmetros, pelo menos, o seu número de identificação (de 1 a 4) e um outro parâmetro que permita determinar o ponto inicial de busca. Podem ser passados quaisquer parâmetros adicionais, conforme a necessidade.

R5. O programa deve realizar $r$ rodadas do seguinte procedimento:

i. `main()` escolhe um número aleatório entre 1 e $m \times n$;
ii. as *threads* buscam (em paralelo) o número sorteado na matriz;
iii. a primeira *thread* a encontrar o número é considerada a vencedora da rodada.

R6. Cada *thread* deve buscar o número sorteado por `main()` partindo de um canto diferente da matriz ($a_{11}, a_{1n}, a_{m1}, a_{mn}$) e percorrendo todos os elementos até encontrar o número procurado. Como a matriz contém todos os números entre 1 e $m \times n$, o número sorteado sempre será encontrado.

R7. Todas as *threads* devem iniciar a busca ao mesmo tempo, depois que o número for sorteado por `main()`.

R8. A primeira *thread* que encontrar o número deve inserir o seu número de identificação em uma lista com a vencedora de cada rodada (i.e., o $k$-ésimo elemento da lista será a *thread* mais rápida da $k$-ésima rodada). Essa lista pode ser estática ou dinâmica.

R9. Depois de encontrar o número sorteado, uma *thread* deve bloquear até que outro número seja sorteado, ou, se for a última rodada, até que a última *thread* termine o seu processamento. Em outras palavras, uma *thread* só pode passar para a próxima rodada (ou finalizar) quando todas as *threads* tiverem concluído a busca. O programa principal (`main()`) deve esperar que todas as *threads* terminem a rodada $k$ antes de avançar para a rodada $k + 1$.

R10. Ao final das $r$ rodadas, o programa principal deve mostrar o número de rodadas em que cada *thread* foi a mais rápida, e declarar a(s) vencedora(s).

R11. O programa deve tratar condições de disputa no código.

O exemplo a seguir ilustra o formato esperado para a saída do programa (invocado com $m = 30$, $n = 40$ e $r = 10$) quando as *threads* 1 e 3 foram as mais rápidas em duas rodadas cada uma, e as *threads* 2 e 4 foram mais rápidas em três rodadas cada.

```
$ ./prog 30 40 10
  thread   1 =>      2 vitorias
  thread   2 =>      3 vitorias
  thread   3 =>      2 vitorias
  thread   4 =>      3 vitorias
------------------------------
Thread(s) vencedora(s): 2 4
```