Tópico: **Listas, Filas e Pilhas**

1. A notação pós-fixa é aquela em que os operadores de uma expressão aritmética são escritos após os operandos. Uma das vantagens desta notação é não precisar de parêntesis para indicação de prioridade das operações. Muitos compiladores fazem transformação da notação infixa para pós-fixa antes de avaliar as expressões aritméticas que aparecem em programas. Por exemplo, a expressão $(3 + 4) \times 2$ equivale à expressão $3\,4 + 2 \times$ na notação pós-fixa.

   Uma pilha pode ser usada para avaliar expressões na notação pós-fixa. Iniciando com uma pilha vazia, varre-se a expressão pós-fixa da esquerda para a direita. Cada vez que um argumento é encontrado, ele é empilhado. Quando um operador é encontrado, é feito o desempilhamento na pilha duas vezes, correspondendo aos operandos e então o operador é aplicado aos operandos desempilhados, e o resultado é empilhado na pilha. A figura abaixo mostra o processamento para a expressão anterior. Aqui, a letra ϵ indica pilha vazia, e o topo é sempre o número mais à direita.

Símbolo processado	Pilha	Operação
—	ϵ	—
3	3	push 3
4	3 4	push 4
+	ϵ	pop 4; pop 3; calcula $7 = 3 + 4$
—	7	push 7
2	7 2	push 2
$\times$	ϵ	pop 2; pop 7 calcula $14 = 7 \times 2$
—	14	push 14

   Com base na descrição acima e usando o conceito de pilha, implemente um programa em C para avaliar uma expressão aritmética na notação pós-fixa, em que são aceitos somente números inteiros na entrada e os operadores +, -, $\times$ e / (divisão inteira).

2. Implementar um programa em C usando o conceito de pilha para verificar o balanceamento de parêntesis em expressões aritméticas. Exemplos de expressões balanceadas: $(a + b) \times ((c + d) - 5)$ e $(((a - b) \times c) - (7 + d))$. Exemplos de expressões desbalanceadas: $(((a + b) - c)$ e $(5 + a)) + 7$.

3. Implementar um jogo de dominó para ser jogado com o computador, em que cada uma das 28 peças é representada através de um par de inteiros, variando de 0 a 6: (0,0), (0,1), ..., (0,6), (1,1), (1,2), ..., (1,6), ..., (6,6). Use listas encadeadas para representar as peças do usuário, do computador, do monte e as dispostas na mesa.

4. Implementar um simulador de escalonador de processos que implementa uma fila circular e funciona de acordo com as regras abaixo:

   - inicialmente o usuário informa o número de processos
   - em seguida informa para cada processo o tempo total de execução, medido em segundos
   - e por último informa a fatia de tempo (*timeslice*)

   Após a entrada dos dados anteriores, o programa simula a execução dos processos, dando 1 fatia de tempo para cada um, após a qual o processo em execução "vai para o final da fila". O programa deve exibir o conteúdo da fila a cada troca de processo mostrando o tempo restante para cada processo, até que todos sejam finalizados.

5. (**Desafio!**) O jogo *FreeCell* é um jogo de cartas que tem por finalidade agrupar todas as cartas do mesmo naipe em 4 pilhas em ordem crescente. Antes de iniciar o jogo, as cartas são embaralhadas e distribuidas na mesa em 8 pilhas. A figura a seguir mostra uma configuração inicial.

No canto direito superior, tem-se 4 pilhas em que se deve colocar as cartas agrupadas por naipe em ordem crescente.

Os 4 espaços no canto esquerdo superior são usados como espaços de trabalho em que se pode colocar apenas 1 carta de qualquer uma das pilhas de cartas embaralhadas.

As cartas embaralhadas das 8 pilhas só podem ser movimentadas segundo a definição de pilha. E além disso, ao se movimentar uma carta de uma pilha para outra, os valores devem ser decrescentes e de naipes de cores alternadas.

Implemente uma versão do jogo, considerando:

- um baralho padrão de 52 cartas de 4 naipes (paus, copas, espada e ouros)
- uma carta é identificada pelo naipe e por seu valor: (A,1,2,3,4,5,6,7,8,9,10,J,Q,K)
- crie uma estrutura de dados adequada para representação/identificação das pilhas existentes no jogo