EXERCÍCIOS PILHA

1) Implementar o tipo PilhaDupla de números reais, que permite a utilização de duas pilhas simultaneamente. As pilhas são identificadas pelos números 1 e 2.

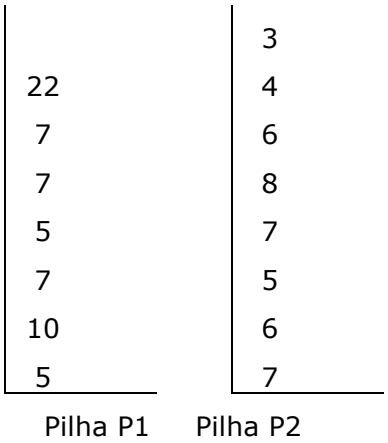
```
/* TAD PilhaDupla de números */
/* As duas pilhas compartilham a mesma estrutura de armazenamento de dados */
/* o array tabela armazena a pilha 1 a partir da posição 0, seguindo para a posição 1*/
/* e armazena a pilha 2 a partir da posição Max-1, seguindo para a posição Max-2 */
        #define Max 20
        typedef struct {
             int topo1;
             int topo2;
             float tabela[Max];
        } PilhaD;

/* construtor  -  cria uma pilha dupla vazia */
    void criarPilhaVazia(PilhaD *);
/* acesso   -  o parâmetro do tipo int indica qual das duas pilhas (1 ou 2)deve ser acessada */
    float acessarTopo(PilhaD *, int);
    bool verificarPilhaVazia(PilhaD, int);
    bool verificarTemEspaço(PilhaD, int);
/* manipulação  -  o parâmetro do tipo int indica qual das duas pilhas deve ser alterada */
    void push(PilhaD *, int, float);
    void pop(PilhaD *, int);
```

Ilustração: A pilha dupla P contém duas pilhas de números. A próxima posição para a pilha P1 é a posição 7 e a próxima posição para a pilha P2 é a posição 11.

tabela

5	10	7	5	7	7	22						3	4	6	8	7	5	6	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

topo1 | 7 | topo2 | 11 |

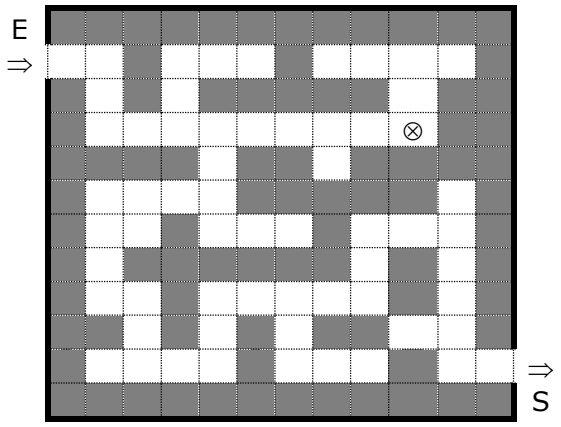


2) Considere disponível o TAD Pilha, conforme a interface a seguir. Implemente as funções copiarPilha e inverterPilha.

/* Essa função serve para copiar todo o conteúdo de uma pilha A (a primeira pilha) em uma segunda pilha B */
/* A ordem dos elementos na pilha B deve ser igual à ordem dos elementos da pilha A */
void copiarPilha(Pilha *, Pilha *);
/* Essa função modifica o conteúdo da pilha, invertendo sua ordem dentro da mesma */
void inverterPilha(Pilha *)

/* TAD Pilha – uma pilha de elementos do tipo Item */
/* construtores */
void criarPilhaVazia(Pilha *);
/* acesso */
Item acessarTopo(Pilha *);
bool verificarPilhaVazia(Pilha);
bool verificarPilhaCheia(Pilha);
/* manipulação */
void push(Pilha *, Item);
void pop(Pilha *);

3) Um labirinto pode ser representado por uma matriz nxn de bits. A entrada do labirinto é na posição (e,1) e a saída do labirinto é na posição (s,n). Em cada casa do labirinto, identificada por (i,j) na matriz, há quatro possíveis sentidos de movimentos: 1- para o norte, 2- para o leste (direita), 3- para o sul e 4- para o oeste (esquerda). Um caminho dentro do labirinto pode ser representado por uma seqüência como 23322222222 (ou LSSLLLLLLLL), que leva à casa marcada com ⊗.



A=

1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	0	0	0	0	1
1	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	0	0	0	0	0	0	1	1
1	1	1	1	0	1	1	0	1	1	1	1
1	0	0	0	0	1	1	1	1	1	0	1
1	0	0	1	0	0	0	1	0	0	0	1
1	0	1	1	1	1	1	1	0	1	0	1
1	0	0	1	0	0	0	0	0	1	0	1
1	1	0	1	0	1	0	1	1	0	0	1
1	0	0	0	0	1	0	0	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1

Dada a matriz A e os valores n,e,s (inteiros positivos, com $1 \leq e,s \leq n$), determinar um caminho da casa (e,1) à casa (s,n).

Ajuda: Cada movimento provoca uma alteração em um dos dois valores (i, j) que identificam uma casa no labirinto. Na casa corrente (onde você está) é preciso examinar as 4 possibilidades de movimento (1,2,3 ou 4) até encontrar um movimento que leve a uma casa ainda não visitada. Chegando a um "beco", uma casa sem saída, é preciso voltar pelo mesmo caminho percorrido (como se tivesse um fio marcando de onde você veio) e daí, tentar avançar novamente. Para identificar uma casa já visitada, é conveniente ter uma matriz auxiliar V, de ordem n, para marcar as casas já atingidas. Para lembrar de onde você veio e poder voltar atrás, use uma pilha, guardando a localização e o sentido do movimento (uma terna de números i, j, k).

Esboço
Inicializações; casa ← (e,1); colocar (e,1,0) na pilha;

```
repita
   (i,j,k) ← acessar topo da pilha;
   enquanto  (k < 4) e (não achou)  faça
      k ← k + 1;     obter nova casa (g,h);
      se casa = saída
         então achou ← true
         senão se (casa válida)
                  então    marcar a nova casa (g,h)
                           colocar (i,j,k) na pilha
                           k ← 0; (i,j) ← (g,h);
   popPilha
até que achou;
esvaziar a pilha para obter o caminho.
```

4) Considere o estacionamento de trens (figura 1) como uma pilha, usado para trocar a ordem dos vagões que estão no trilho da direita. Os vagões são numerados de 1 a n e desejamos mudar essa ordem para que saiam pelo trilho da esquerda. Um vagão do trilho da direita pode entrar na pilha-estacionamento, pode ficar na pilha e pode sair somente para o trilho da esquerda. Não pode voltar da pilha para o trilho da direita. Por exemplo, se n = 3 e os vagões chegam na ordem 1,2,3 (no trilho da direita),pode ocorrer: o vagão 1 entra na pilha e em seguida entra o vagão 2 na pilha. Sai o vagão 2 da pilha, em seguida sai o vagão 1 da pilha e o vagão 3 passa direto da direita para a esquerda, ficando os vagões na ordem 2,1,3 no trilho da esquerda.

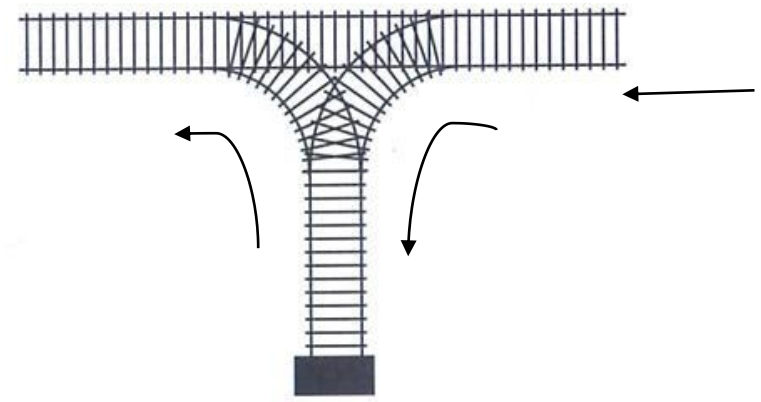


Figura 1

a) Quais as possíveis permutações que podemos obter com n = 3? Qual a seqüência de movimentos? (pop = retira um vagão da pilha para o trilho da esquerda, push = coloca um vagão do trilho da direita na pilha, move um vagão da direita para a esquerda)

b) Considerando pop=R, push =C, move =M, n = 10, a seqüência CCRCCRRRMM é uma seqüência de operações válidas? Qual o resultado final no trilho da esquerda? E a seqüência CCMMRRRCCR?

c) Quais as condições para que uma cadeia de n caracteres pertencentes a {C, R, M} seja uma seqüência de operações válidas?