UNIVERSIDADE FEDERAL DE SANTA CATARINA (UFSC)
CAMPUS JOINVILLE

Departamento de Engenharias da Mobilidade

Especificação

Uma empresa possui um único guindaste que movimenta containers em seu pátio. Os donos da empresa acreditam que os salários pagos aos operadores são muito altos e querem substitui-los por um sistema automatizado, visando reduzir os custos. O sistema robótico de movimentação já foi implantado. Cabe a você desenvolver o sistema lógico de movimentação (na verdade, uma simulação do mesmo).

Seu programa deve assumir que, inicialmente, todos containers encontram-se alinhados no pátio da empresa, um ao lado do outro, tal qual foram posicionados ao chegar. A figura abaixo ilustra o estado inicial dos containers, a partir do qual seu programa deverá permitir manipulações. Note que cada container possui um identificador único que vai de 0 à *n-1*, sendo *n* o número de containers. Esses identificadores fornecem também qual a posição inicial do container (ela será necessária em alguns casos).

0	*1*	*2*	*3*	*4*	*5*	*...*	*n-3*	*n-2*	*n-1*

A partir deste estado inicial, seu programa deverá permitir as seguintes movimentações:

- `mova a sobre b`
  Onde *a* e *b* são números de containers. Esta movimentação coloca o container identificado por *a* exatamente sobre o container identificado por *b*. Antes, porém, quaisquer containers que estiverem sobre *a* e *b* são retornados às suas posições originais no pátio da empresa.

- `mova a topo b`
  Onde *a* e *b* são números de containers. Esta movimentação coloca o container identificado por *a* no topo da pilha de containers que contém o container identificado por *b*. Antes, porém, quaisquer containers que estiverem sobre *a* são retornados a sua posição inicial.

- `empilhe a sobre b`
  Onde *a* e *b* são números de containers. Movimenta o container *a* e quaisquer containers que estiverem acima de *a* (ou seja, a pilha de containers definida a partir de *a*) para a posição exatamente acima do container *b*. Antes, porém, quaisquer containers que estiverem sobre *b* devem ser retornados às suas posições originais no pátio da empresa. Os containers que estiverem acima de *a* devem manter a ordem inicial em que estavam.

- `empilhe a topo b`
  Onde *a* e *b* são números de containers. Movimenta o container *a* e quaisquer containers que estiverem acima de *a* (ou seja, a pilha de containers definida a partir de *a*) acima do topo da pilha que contém o container *b*. Os containers que estiverem acima de *a* devem manter a ordem inicial em que estavam, assim como os containers abaixo e acima de *b*.

- `sair`
  Encerra a execução do programa e mostra o estado das pilhas de containers.

Seu programa deve ler, a partir do teclado, um inteiro *n* que especifica o número total de containers que se encontram no pátio da empresa, inicialmente dispostos lado a lado, tal qual especificado anteriormente. Você pode assumir que $0 < n < 2500$. Após a leitura deste valor, cada linha de entrada deverá conter um dos comandos especificados acima. Seu programa deve aceitar entradas até que o comando *sair* seja encontrado. Neste momento, seu programa deverá imprimir o estado atual das pilhas de containers, *exatamente como descrito no exemplo que será fornecido*. Você pode assumir que *todos* os comandos fornecidos seguirão o padrão especificado acima. Não serão fornecidos números de containers inválidos. Além disso, seu programa deve desconsiderar quaisquer comandos para os quais *a* e *b* sejam iguais ou se encontrem em uma mesma pilha. Nestes casos seu programa não deve fazer nada, prosseguindo para os demais comandos da entrada.

A saída do seu programa deverá mostrar o estado final dos containers no pátio da empresa. As posições iniciais são numeradas de 0 à *n*, onde *n* corresponde ao número de containers. Você deve imprimir uma posição por linha, seguida do caracter dois pontos (":"). Se existirem containers na posição da linha corrente, você deve imprimir um espaço após os dois pontos. Após o espaço, imprima o identificador de cada container que está na posição, separando identificadores por exatamente um espaço em branco. Não coloque nenhum espaço em branco após a impressão do último container da linha. Após a impressão de cada linha você deve imprimir uma quebra de linha (após a impressão da última linha imprima uma única linha nova). Exemplos de entrada e saída são fornecidos abaixo.

Entrada 1	Entrada 2
10 mova 9 sobre 1 mova 8 topo 1 mova 7 topo 1 mova 6 topo 1 empilhe 8 topo 6 empilhe 8 topo 5 mova 2 topo 1 mova 4 topo 9 sair	21 mova 2 sobre 1 mova 3 sobre 2 mova 4 sobre 3 mova 5 topo 1 empilhe 1 topo 10 mova 9 topo 8 mova 11 topo 8 empilhe 3 topo 8 empilhe 8 topo 3 mova 20 topo 19 empilhe 19 topo 18 empilhe 18 sobre 15 mova 15 topo 3 empilhe 20 sobre 19 empilhe 19 sobre 18 empilhe 18 topo 17 sair
Saída 1	Saída 2
0: 0 1: 1 9 2 4 2: 3: 3 4: 5: 5 8 7 6 6: 7: 8: 9:	0: 0 1: 2: 3: 4: 5: 6: 6 7: 7 8: 8 9 11 3 4 5 15 9: 10: 10 1 2 11: 12: 12 13: 13 14: 14 15: 16: 16 17: 17 18 19 20 18: 19: 20:

Para este exercício é mandatório o uso do TAD Pilha, implementado anteriormente. Organize seu código, colocando o as declarações do TAD Pilha no arquivo pilha_interface.h e sua implementação em pilha.c. Você pode copiar as implementações do VPL anterior, no qual você implementou o TAD Pilha. O uso do TAD Lista é opcional. Caso você opte por usá-lo, organize seu código de forma similar.

A dinâmica desse laboratório segue o estilo dos laboratórios anteriores.

Avaliação:

Uso de variável global: -5
Um vazamento de memória: -20%
Mais de um vazamento de memória: -40%
Não compila: 0.
Expirou o tempo de 15 segundos: 0.
Erro de execução (fim de programa anormal): 0.
Erro com ponteiro inválido: 0.
Uso da função exit: 0.
Obs: não utilize em seu arquivo .c caracteres especiais (á,é,õ,ã, etc) e comentários de múltiplas linhas iniciando e terminando em linhas diferentes.

Toda as operações do TDA numero_grande_t possuem o mesmo peso na avaliação.

Para comentar múltiplas linhas siga o exemplo abaixo:

```
//linha 1 de um comentario
//linha 2 de um comentario
//linha 3 de um comentario
```

ou

```
/*linha 1 de um comentario */
/*linha 2 de um comentario */
/*linha 3 de um comentario */
```