Instituto de Computação
Curso de Sistemas de Informação
Estruturas de Dados (TCC00171)

Lista de Exercícios 4

Para a solução dos exercícios desta lista, considere um tipo abstrato* Pilha *que armazena valores inteiros e permite a recuperação destes valores na ordem inversa, e um tipo abstrato* Fila *que armazena números inteiros e permite a recuperação destes valores na ordem em que foram armazenados. As interfaces* pilha.h *e* fila.h *estão definidas conforme descrito a seguir.

pilha.h

```
typedef struct pilha Pilha
Pilha* pilha_cria(void)
void pilha_push(Pilha *p, int x)
int pilha_pop(Pilha *p)
int pilha_cheia(Pilha *p)
int pilha_vazia(Pilha *p)
void pilha_libera(Pilha *p)
```

fila.h

```
typedef struct fila Fila
Fila* fila_cria(void)
void fila_insere(Fila *p, int x)
int fila_retira(Fila *p)
int fila_cheia(Fila *p)
int fila_vazia(Fila *p)
void fila_libera(Fila *p)
```

1) Escreva as funções necessárias para implementar um tipo abstrato *Pilha* que armazena números inteiros, utilizando como estrutura de armazenamento uma lista encadeada circular.

2) Escreva as funções necessárias para implementar um tipo abstrato *Fila* que armazena números inteiros, utilizando como estrutura de armazenamento uma lista encadeada circular.

Para a solução dos exercícios 2 a 10, considere que as pilhas ou filas só podem ser utilizadas através das funções definidas nas interfaces dos tipos abstratos, ou seja, não deve ser feita nenhuma suposição a respeito das funções que implementam esses tipos abstratos.

3) Escreva uma função que receba ponteiros para duas pilhas e verifica se estas são iguais, isto é, se contêm os mesmos valores armazenados na mesma ordem. A função deve retornar 1, se as pilhas são iguais, ou 0, caso contrário. O protótipo da função é:

```
int pilhas_iguais(Pilha *p1, Pilha *p2);
```

4) Escreva uma função que receba ponteiros para duas filas e verifica se estas são iguais, isto é, se contêm os mesmos valores armazenados na mesma ordem. A função deve retornar 1, se as filas são iguais, ou 0, caso contrário. O protótipo da função é:

```
int filas_iguais(Fila *f1, Fila *f2);
```

5) Escreva uma função recebe o ponteiro para uma pilha e imprime seus elementos na mesma ordem em que foram empilhados. Pode ser utilizada uma pilha auxiliar. O conteúdo da pilha fornecida como parâmetro não deve ser alterado. O protótipo da função é:

```
void imprime_pilha(Pilha* p);
```

6) Escreva uma função que testa o funcionamento de uma implementação de pilha. A função recebe como parâmetro um vetor de inteiros, deve criar uma pilha, inserir todos os elementos do vetor na pilha e depois retirá-los, verificando se são apresentados na ordem correta. A função deve retornar 1, se a implementação funciona corretamente, ou 0, caso contrário. O protótipo da função é:

```
int testa_pilha(int *v, int n);
```

7) Escreva uma função que testa o funcionamento de uma implementação de fila. A função recebe como parâmetro um vetor de inteiros, deve criar uma fila, inserir todos os elementos do vetor na fila e depois retirá-los, verificando se são apresentados na ordem correta. A função deve retornar 1, se a implementação funciona corretamente, ou 0, caso contrário. O protótipo da função é:

```
int testa_fila(int *v, int n);
```

8) Escreva uma função que recebe o ponteiro para uma fila e, utilizando uma pilha, inverte os elementos desta fila. O protótipo da função é:

```
void inverte_fila(Fila *f);
```

9) Uma cadeia de caracteres contendo apenas os caracteres ‘(’ e ‘)’ é armazenada como um vetor de inteiros. Escreva uma função que, utilizando uma pilha, verifica se a cadeia de caracteres está balanceada, ou seja, para cada “abre parêntesis”, há um “fecha parêntesis”. A função deve retornar 1, se a cadeia é balanceada, ou 0, caso contrário. O protótipo da função é:

```
int parentesis_balanceados(int *v, int n);
```

10) Implemente uma função recursiva que receba uma pilha como parâmetro e retorne como resultado uma cópia dessa pilha. A pilha recebida como parâmetro não deve ter seu conteúdo original modificado. O protótipo da função é:

```
Pilha* copia_pilha(Pilha *p);
```