Lista de Exercícios 1

Considerando a implementação de um tipo abstrato Pilha de caracteres, cuja interface está definida no arquivo *pilha.h* e a implementação no arquivo *pilha.c* apresentada em sala de aula, implemente as questões abaixo.

1. **Implemente uma função que receba duas pilhas, p_1, p_2, e passe todos os elementos da pilha p_2 para o topo da pilha p_1. Essa função deve obedecer ao protótipo:**

   ```
   void concatena(Pilha* p1, Pilha* p2);
   ```

   Solução

   ```
   /**
    * Concatena o conteúdo da Pilha p2 para a Pilha p1,
    * colocando-a no topo da pilha p1.
    */
   void concatena(Pilha* p1, Pilha* p2)
   {
        while (!empty(p2))
        {
           char c = pop(p2);
           push(p1,c);
        }
   }
   ```

2. **Implemente uma função que receba uma pilha e um caractere como parâmetros e retorne 1 se o caractere informado fizer parte da pilha ou 0 caso contrário. Essa função deve obedecer ao protótipo:**

   ```
   int pesquisa(Pilha* p, char c);
   ```

   Solução

   Para implementar essa função, sem modificar a pilha recebida, devemos comparar cada elemento do vetor da pilha, com o valor que queremos verificar a sua existência na pilha.

   ```
   /**
    * Pesquisa pelo elemento c na Pilha p. A função retorna 1
    * se o elemento c estiver na pilha ou 0 caso contrário.
   ```

```
 */
int pesquisa(Pilha* p, char c)
{
    int i = p->topo;
    while (i >= 0 && p->vet[i] != c)
       i--;

   return i < 0 ? 0 : 1;
}
```

3. **Implemente uma função que receba uma pilha como parâmetro e retorne como resultado uma pilha com os elementos na ordem inverta da pilha de entrada. A figura 1 ilustra essa inversão.**

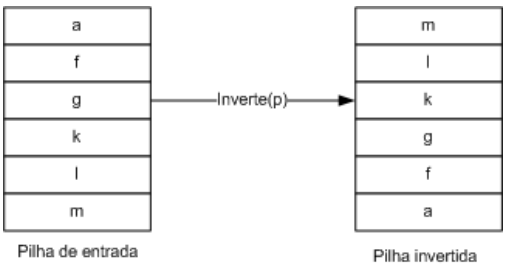

Figura 1: Exemplo de pilha invertida

Essa função deve obedecer ao protótipo:

```
Pilha* inverte(Pilha* p);
```

Solução

De modo geral, para invertemos uma pilha qualquer, basta desempilhar cada elemento da pilha recebida como parâmetro e empilhá-lo na pilha de retorno. Para não modificar a pilha de entrada, será necessário criar uma cópia e desempilhar os elementos da cópia.

```
/*
 * Cria uma Pilha com os elementos na ordem inversa aos
 * da pilha de entrada.
 */
Pilha* inverte(Pilha* p)
{
   Pilha* copy = copia(p);
   Pilha* inv = create();
```

```
  while (!empty(copy))
  {
    push(inv,pop(copy));
  }
  libera(copy);
  return inv;
}
```

4. **Implemente uma função que receba uma pilha como parâmetro e retorne como resultado uma cópia dessa pilha. Essa função deve obedecer ao protótipo:**

   ```
   Pilha* copia(Pilha* p);
   ```

   Solução

```
/*
 * Cria e retorna uma cópia da pilha p recebida como parâmetro.
 */
Pilha* copia(Pilha* p)
{
  Pilha* copia = create();
  copia->topo = p->topo;
  int i;
  for(i = 0; i < p->topo; i++)
  {
    copia->vet[i] = p->vet[i];
  }
  return copia;
}
```