Second written examination of

Algoritmos e Estruturas de Dados

Outubro 26, 2015 Duration: no more than 30 minutes

Name:

Student number:

5.0 **1:** **What is a stack? What operations does it provide?**

Answer:

Uma pilha (*stack*) é uma estrutura de dados para a qual as operações realizadas seguem uma ordem específica. Essa ordem é LIFO (*Last In First Out*), o que significa que se quisermos, por exemplo, remover um elemento da pilha, devemos sempre retirar o último elemento que lá foi colocado.
A pilha tem seis operações básicas:
- *push*: colocar um novo elemento na pilha; o elemento é colocado no topo da pilha; se a pilha já estiver cheia, trata-se de uma situação de *overflow*
- *pop*: retirar um elemento da pilha; o elemento é retirado do topo da pilha; se a pilha já estiver vazia, trata-se de uma situação de *underflow*
- *peek* ou *top*: retorna o elemento que está, nesse momento, no topo da pilha
- operação de criação da pilha
- operação de destruição da pilha
- operação que indica o tamanho da pilha

3.0 **2:** **Write on the left C++ code that implements the pop function (stack implemented as an array, data items are of type T), using some of the lines of code presented on the right.**

```
{
```

[C++ não sai no teste]

```
}
```

```
T pop(void)
void pop(T v)
assert(cur_size > 0);
assert(cur_size < max_size);
return data[cur_size++];
return data[cur_size--];
return data[++cur_size];
return data[--cur_size];
data[cur_size++] = v;
data[cur_size--] = v;
data[++cur_size] = v;
data[--cur_size] = v;
```

5.0 3: What is a queue? What operations does it provide?

Answer:

Uma fila (*queue*) é uma estrutura de dados cujas operações são efetuadas numa ordem específica. Essa ordem é FIFO (*First In First Out*), o que significa que se quisermos, por exemplo, retirar um elemento da fila, devemos sempre retirar o primeiro que lá foi colocado (ou seja, o elemento que já foi colocado na fila há mais tempo, de entre os disponíveis, e que se trata sempre do elemento no fim/fundo da fila).
A fila tem cinco operações básicas:
- *enqueue*: colocar um novo elemento na fila; o elemento é colocado no topo da fila; se a fila já estiver cheia, há uma situação de *overflow*
- *dequeue*: retirar um elemento da fila; o elemento é retirado do fundo da fila; se a fila estiver vazia, há uma situação de *underflow*
- operação de criação da fila
- operação de destruição da fila
- operação para determinar o tamanho da fila

3.0 4: Explain how to increment an index in a circular buffer.

Answer:

Um *buffer* circular é um *array* em que a aritmética envolvida nos índices é feita de acordo com o tamanho do *array*. Quando queremos incrementar um ponteiro para um dado índice do *buffer* circular, se o índice para o qual o ponteiro aponta antes da incrementação for menos que *array_size* - 1, então basta somar 1 ao índice; caso contrário, o índice seguinte é 0. Isto garante a circularidade do *buffer*

4.0 5: Write a C function that counts the number of nodes of a linked list that appear before a given node n. The function must return -1 if the node does not belong to the linked list. Use some of the following lines of code.

```
for(int c = 0;head != NULL;head = head->next,c++)
for(int c = 0;head != NULL;head = head->next,c--)
for(int c = 0;head != NULL;head = head->prev,c++)
for(int c = 0;head != NULL;head = head->prev,c--)
if(head == n)
if(head != n)
return c;
return c - 1;
return c + 1;
return -1;
```

Answer:

```
int count_before(node *head,node *n)
{     for (int c = 0; head != NULL; head = head->next, c++) {
          if (head==n) {
              return c;
          }
      }
}     return -1;
```