Second written examination of

Algoritmos e Estruturas de Dados

Outubro 24, 2016 **Duration: no more than 40 minutes**

Name:

Student number:

Note that `assert(expression);` terminates a program when `expression` is false and that a binary tree node (question 4) is defined in the following way:

```
typedef struct tree_node
{
  struct tree_node *left;   // pointer to the left branch (a sub-tree)
  struct tree_node *right;  // pointer to the right branch (a sub-tree)
  struct tree_node *parent; // pointer to the paprent node (NULL for the root node)
  int data;                 // the data
}
tree_node;
```

5.0 **1:** Is it possible to implement efficiently a queue (first in, first out) using a singly-linked list? If so, how? If not, why not?

Answer:

5.0 **2:** Describe how is information stored in a max-heap. Illustrate your explanation with an example (in the example the max-heap contents should be 1, 2, 3, 4, 5, and 6).

Answer:

4.0 **3:** The following code can be used to put a data item (in this case an integer) in a circular buffer. Explain how it works.

```
#define BUFFER_SIZE 1024

int buffer[BUFFER_SIZE];
int read_index = 0,write_index = 0;

void put(int data)
{
  buffer[write_index] = data;
  write_index = (write_index + 1 < BUFFER_SIZE) ? write_index + 1 : 0;
  assert(read_index != write_index);
}
```

Answer:

6.0 **4:** The following two functions are supposed to check if an ordered binary tree is indeed ordered. Which one is the correct one? Why?

```
void check_node_v1(tree_node *link)
{ // use check_node_v1(root) to check an entire ordered binary tree
  if(link != NULL && link->left != NULL)
  {
    assert(link->data >= link->left->data && link == link->left->parent);
    check_node_v1(link->left);
  }
  if(link != NULL && link->right != NULL)
  {
    assert(link->data <= link->right->data && link == link->right->parent);
    check_node_v1(link->right);
  }
}

void check_node_v2(tree_node *link,tree_node *parent,int min_bound,int max_bound)
{ // use check_node_v2(root,NULL,INT_MIN,INT_MAX) to check an entire ordered binary tree
  if(link != NULL)
  {
    assert(min_bound <= link->data && link->data <= max_bound && link->parent == parent);
    check_node_v2(link->left,link,min_bound,link->data);
    check_node_v2(link->right,link,link->data,max_bound);
  }
}
```

Answer: