Data Structures and Algorithms – 2018

Assignment 1 – 0% of Continuous Assessment Mark

Deadline : 5pm Monday 29th January, via Canvas

The University's *Code of Practice on Assessment and Feedback* says "Formative feedback should be provided on the first piece of work of a particular type in a programme/module". For that reason, this first assignment will be marked (with appropriate additional feedback) as usual, but the mark will not contribute towards the final mark awarded for this module.

Question 1 (10 marks)

You need to insert the numbers 2, 4, 3, 7, one at a time in that order into to an initially empty queue.

Represent that process using the standard constructors `push` and `EmptyQueue`.

Show, in the standard two-cell notation, the resulting queue.

What is the result of the operation `top` on that queue?

What is the result of the operation `pop` on the original queue you created?

What is the result of the operation `pop` followed by `pop` followed by `top` on the original queue you created?

Question 2 (18 marks)

In the lecture notes (Section 3.2) a recursive procedure `last(L)` was defined that returns the last item in the given list `L`. By making the simplest possible modification to that procedure, create a recursive procedure `secondlast(L)` that returns the second to last item in a given list `L`.

What is the time complexity of your algorithm?

Now carry out a more general modification of the `last(L)` procedure to give a recursive procedure `getItem(i,L)` that returns the `i`th item in the given list `L`, where `i` is an integer greater than zero. [Hint: See Lecture Notes Section 6.8.]

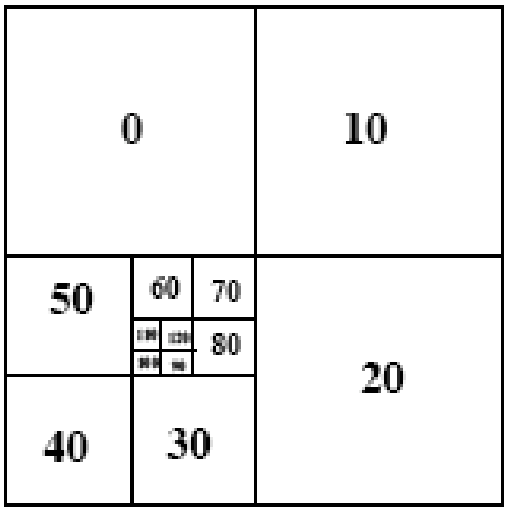
Question 3 (10 marks)

Often one needs to check whether two given lists are the equal, i.e. contain the same items in the same order. Write a recursive procedure `equalList(L1,L2)` that returns `true` if the two given lists `L1` and `L2` are the same, and `false` if they are not. The only procedures it may call are the standard primitive list operators `first`, `rest` and `isEmpty`.

What is the time complexity of your algorithm?

Question 4 (16 marks)

A *quadtree* was defined in the lectures in terms of primitive constructors `baseQT(value)` and `makeQT(luqt,ruqt,llqt,rlqt)`, selectors `lu(qt)`, `ll(qt)`, `ru(qt)` and `rl(qt)`, and condition `isValue(qt)`. Suppose a gray-scale picture is represented by such a quadtree with values in the range 0…255, for example:



Write a procedure `flip(qt)`, that uses the above primitive quadtree operators, to flip the picture about the vertical line through its centre.

Write another procedure `avevalue(qt)`, that uses the above primitive quadtree operators, to return the average gray-scale value across the whole picture.

Question 5 (12 marks)

It is often important to know whether two given binary trees are the identical. Write a recursive procedure `equalBinTree(bt1,bt2)` which returns `true` if the given binary trees `bt1` and `bt2` are the same, and `false` otherwise. You can assume that you have access to the standard primitive binary tree procedures `root(bt)`, `left(bt)`, `right(bt)` and `isempty(bt)`. [Hint: Remember that you can only directly test the equality of numbers, e.g. node values.]

What is the time complexity of your algorithm?

Question 6 (16 marks)

Suppose you have access to the primitive binary tree procedures `root(bt)`, `left(bt)`, `right(bt)` and `isempty(bt)`. Write a procedure `isLeaf(bt)` using them that returns `true` if the binary tree `bt` is a leaf node, and `false` if it is not.

Then write a recursive procedure `numLeaves(bt)` that returns the number of leaves in the given binary tree `bt`. It is only allowed to call the above primitive binary tree procedures and your `isLeaf(bt)` procedure.

Question 7 (18 marks)

How many different orderings of the four numbers {1, 2, 3, 4} are there?

By considering all those possible orderings, draw all possible binary search trees of size four with nodes labeled by the four numbers {1, 2, 3, 4}. After discarding any duplicate trees, how many different binary search trees of size four are there?

For each different tree, state its height, how many leaf nodes it has, and whether it is perfectly balanced.