Foundations of Computer Science – Supervision 2

Question 1: What are exceptions, and why do we use them?

Question 2: (a) Your friend suggests the following modification for quicksort: Instead of choosing one pivot, choose two and partition the list into three lists:

- The elements smaller than both pivots
- The elements bigger than both pivots
- The elements between the two

Implement this variant of the algorithm.

(b) Comment on the approach in part (a). Is it correct? Is it faster than regular quicksort?

(c) Implement an algorithm that takes a list of integers and an integer k, and returns the kth smallest element. Aim for something faster than sorting algorithms. State the average and worst-case time complexity of your algorithm.

Question 3: (a) Tower of Hanoi is a mathematical game. The game consists of three rods and n disks, with diameters $1, 2, \ldots, n$. Initially, all disks are stacked in one rod in order (the smallest one on top). The goal is to move them all to the second rod, with the following conditions:

- In one move, you can pick up one disk from the top of one rod and place it on the top of another one.
- At all times, a larger disk is never on top of a smaller one.

Create a type to represent the rods of the game.

(b) Create a datatype to describe a move in the game. Use good coding practices.

(c) Implement a function that takes the value n and returns a list of moves that move all disks from the first rod to the second rod while respecting the rules of the game.

Question 4: 2009 Paper 1 Question 2 questions b, c.

Question 5: (a) In this task, you will be coding some operations for integer binary search trees. Assume the following type for the tree:

```
type bst = Branch of int * bst * bst | Leaf
```

Implement a function that inserts a new key into a binary search tree.

(b) Implement a function that takes two binary search trees so that all keys in the second tree are bigger than the keys in the first tree, and merges them into one binary search tree.

(c) Implement a function that removes a key from the tree.

(d) Implement a function `split` that takes a binary search tree and a value n and splits it into two binary search trees, so that the first tree consists of all keys smaller than n, and the second consists of values bigger than (or equal to) n.

Question 6: Can you turn any recursive function into a tail-recursive function? Discuss!
Please don't overcomplicate: you don't need rigorous proofs, or very detailed arguments, just write a short justification to your answer.