Algorithms

Exercise 5: Data structures III

1 Past Tripos questions

1. Explain the terms *amortised analysis* and *aggregate analysis*, highlighting the difference between them. (3 marks)

2. Assume that the following two classes, which implement the standard stack (last-in first-out) and queue (first-in first-out) data structures, are available.

```
class Stack                 class Queue
   void push(Item x)           void enqueue(Item x)
   Item pop()                  Item dequeue()
   Boolean isEmpty()           Boolean isEmpty()
```

   (a) A *Multistack* class is derived from *Stack* with the addition of two methods:

   - a **void** *multipush*(*Itemlist l*) that takes a list l of items and pushes each of the items onto the stack (each action of extracting an item from the list and pushing it onto the stack has constant cost), and
   - a **void** *multipop*(**int** m) that takes an integer m and pops that many items off the stack (raising an exception if there were fewer, but don't worry about that)

   Is it true or false that, given an arbitrary sequence of n *Multistack* operations starting from an empty *Multistack*, each operation in the sequence has amortised constant cost? Justify your answer in detail. (5 marks)

   (b) Provide an implementation of class *Queue* using no other data structures than *Item*, *Boolean*, and *Stack*. The amortised running time of each *Queue* method must be constant. (*Note that you may only use the Stack as a black box: you are not allowed to access its internal implementation.*) (7 marks)

   (c) Using the potential method, prove that the amortised running time of all your *Queue* methods from the previous part is indeed constant. (5 marks)

2 More past questions

3. As part of the Algorithms tick you were asked to implement a heap. Suppose that your implementation was based on an array of size n with indices ranging from 0 to $n-1$. If we attempt to insert an element at position n, we need to allocate a new, larger array and copy all n elements from the old array into the new one. Using the potential method, show what the amortised worst-case time complexity of insertion is. Explain your answer in detail and describe how much bigger you would make the a new array. (10 marks)

4. Originally, this question was about the array-based implementation of binary search trees from two weeks ago. Explain why your answer for the previous question can not trivially be applied to binary search trees. You should reference both, balanced and unbalanced, implementations. (5 marks)