OLLSCOIL NA hÉIREANN, CORCAIGH
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

Examination Session and Year	**SAMPLE PAPER**
Module Code	**CS2515**
Module Title	**Algorithms and Data Structures I**
Paper Number	I
External Examiner	XXX
Head of School/ Department	YYY
Internal Examiners	Professor Ken Brown
Instructions to Candidates	Answer all questions Total Marks: **80**
Duration of Paper	90 minutes
Special Requirements	The use of electronic calculators is permitted

1. (This complete question is worth 30 marks)

(i) Explain using pseudocode and diagrams the structure of a doubly linked list. Describe the representation of a node, giving a clear description of the instance variables.

(5 marks)

(ii) Explain the defining characteristics of the Queue ADT, and state the standard operations it offers.

(4 marks)

(iii) Consider the following sequence of actions, where a letter indicates the addition of an object with that label into a data collection, and '*' represents the removal of an object. The addition of 'p' is the start of the sequence.

*p q * d g* * f a z * * * k m*

Show final state of the data structure when this sequence is processed by
(a) a Stack (implemented using a list)
(b) a Queue (implemented using a list)
where 'removal' is to be interpreted as the appropriate method from the ADT.

(4 marks)

(iv) Suppose that elements to be added to a list have a value, plus a key. The keys are not necessarily unique - that is, two different items in the list could have the same key. Give a detailed design, including pseudocode, for an implementation of a list that maintains the elements in sorted order of their keys - each element of the list should have a key lower than or equal to the element that comes after it. You must specify whether the list is a Python list, or a singly-linked list or a doubly-linked list. [Note that you are not being asked to write a sort algorithm - your lists should always be sorted, so you have to describe how to add a new element (with key and value) into the list in the right position, or remove an item given the ley as input.]

(10 marks)

(v) What is the worst case complexity for adding a new element into your list of part(iv)? What is the worst-case complexity for building a list (as described in (iv) above) of *n* elements step by step from an empty list?

(3 marks)

(vi) In many applications, there is an observed pattern in which items (e.g. files in cloud storage) that have been accessed recently are more likely to be accessed next. To service these applications, we can maintain a list where the elements are sorted in order of the time at which they were last accessed, with the front of the list being the most recently accessed. Explain how you would adapt your implementation of the list in (iv) above to support this ordering.

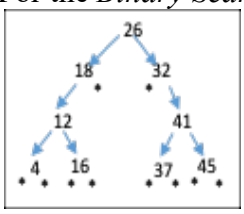
(4 marks)

2. (This complete question is worth 30 marks)

(i) Describe a Binary Search Tree, and give a clear definition of the properties that define it.

(4 marks)

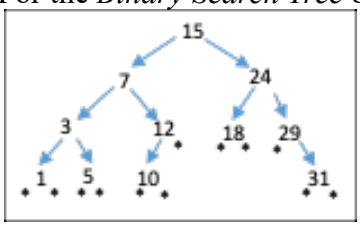
(ii) For the *Binary Search Tree* below, show the result of



(a) adding key 23, and then, from the resulting tree,
(b) removing key 32

(6 marks)

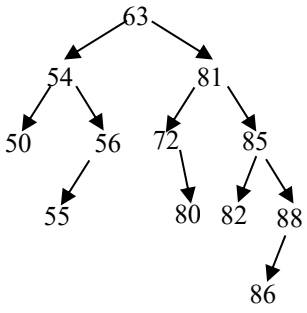
(iii) For the *Binary Search Tree* below, show the result of removing key 15.



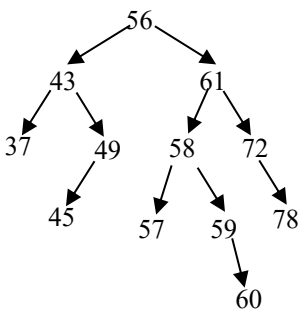
(6 marks)

(iv) For the AVL trees below, show the result of removing 56.

(a)

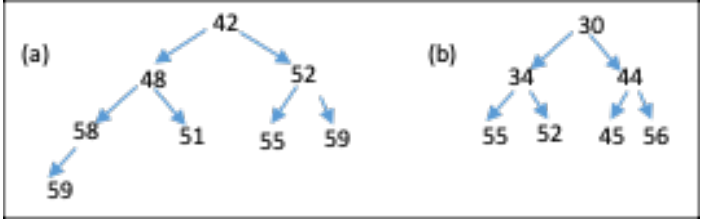


(b)



(8 marks)

(v) The two trees below represent *min Binary Heaps* (and so they are **not** Binary Search Trees). Show the results when you add 44 to the min binary heap in (a), and you remove 30 from the min binary heap in (b):



(6marks)

3. (This complete question is worth 20 marks)

(i) For the following HashTable, which is implemented using *open addressing* with *linear probing*, show the result after each of the following operations is applied in succession (where the result is either a change in the array, or the return of a value):

```
| Ø | Ø |(45,'a')| Ø | Ø | Ø |(12,'b')|(28,c)| Ø | Ø |(57,d)| Ø |
  0   1     2      3   4   5     6        7    8   9    10    11
```

(a) setitem(97, 'r'), where 97 hashes and compresses to 4

(b) setitem(63, 'q') where 63 hashes and compresses to 6

(c) delitem(28), where 28 hashes and compresses to 7

(d) contains(63)

(e) setitem(81, 's'), where 81 hashes and compresses to 6

(5 marks)

(ii) Suppose that we want to implement a map or dictionary, but at any time we could be asked to list the current items in order of their keys. Give a design of this ordered dictionary that is more efficient than the hash table implementation, and justify your answer. Marks will be given for correctness and clarity of your description, and the efficiency of the design.

(8 marks)

(iii) The binary heap implementation of the priority queue is efficient, but it assumes that the priorities of the items never change once they have been added to the PQ. Give a design for how we could adapt or augment the implementation to allow updates of the priorities without compromising on the efficiency. Marks will be given for correctness and clarity of your description, and the efficiency of the design.

(7 marks)

Total: 80 marks

END OF PAPER