University of Glasgow

XXXday May XX, 2016
XX.XX am/pm – XX.XX am/pm
(Duration: X hour XX minutes)

DEGREES OF MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

Algorithms and Data Structures 2

(Answer all 5 questions.)

This examination paper is worth a total of 50 marks

You must not leave the examination room within the first hour or the last half-hour of the examination. (for exams of 2 hours duration)

or

You must not leave the examination room within the first half hour or the last fifteen minutes of the examination. (for exams of less than 2 hours duration)

1. A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the class definition below, Stack is implemented as an array of parameterised type E.

```
public class Stack<E> {

        private int capacity; // stack capacity
        private E[] S;        // E array used to implement the stack
        private int tos = -1; // index for the top of stack

        public ArrayStack(int capacity){
            this.capacity = capacity;
            S = (E[]) new Object [capacity];
        }

        public int size(){...}
        public boolean isEmpty(){...}
        public boolean isFull(){...}

        public void push(E element) throws StackException {...}
        public E top() throws StackException {...}
        public E pop() throws StackException {...}
    }

    public class StackException extends RuntimeException {
      public StackException(){}
      public StackException(String msg){super(msg);}
    }
```

   (a) Give a Java implementation for the undefined methods above (`size`,`isEmpty`, `isFull`, `push`, `top`, `pop`) taking care to address stack overflow and stack underflow. **[8]**

   (b) The stack could have been implemented using a singly linked list (rather than a one dimensional array as above). What would be the advantages and disadvantages of using a linked list rather than an array? **[2]**

   (c) A queue (first-in first-out) can be represented using a pair of stacks. Describe how this might be done. In particular, explain how the enqueue and dequeue operations would be realised, and give the complexity of these operations using the big-O notation. **[8]**

2. A map abstract data structure allows us to store key-value pairs (k,v) , but is different from the dictionary abstract data type in that each key is unique. Consequently the association of keys to values defines a mapping. We might implement the map abstract data type using a hash table.

   (a) What is a hash table and why would we want to use one? **[2]**

   (b) What is a hash function and what properties would we like it to have? **[2]**

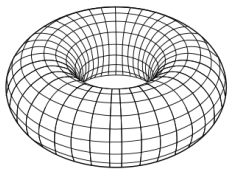
   (c) What is a "collision"? **[1]**

   (d) Describe the "separate chaining" method for hash collision resolution **[2]**

   (e) Describe the "linear probing" method for hash collision resolution, taking into consideration the methods insert, find, and remove **[3]**

3. Describe the QuickSort algorithm for sorting an array of integers into non-decreasing order. Include details of how the pivot might be chosen, the best and worst case complexity of the algorithm and how these complexities can arise. **[8]**

4. In a cellular automata, the "Moore Neighbourhood" of a cell is the eight cells surrounding that cell on a torus represented by an **n** by **n** two-dimensional array, with left and right edges joined and top and bottom edges joined. A torus is shown on the left below and the Moore Neighbourhood of a cell on the right, where the grey cells make up the Moore neighbourhood of the dark cell.



The code snippet below computes the number of occupied cells in the Moore neighbourhood of every cell on the torus, where A[i][j] = 1 when that cell is occupied, zero otherwise. The array element B[i][j] is then the number of occupied cells in the neighbourhood of the cell A[i][j]. Using the big-O notation, what is the complexity of this code? Explain your answer.

```
for (int i=0;i<n;i++)
    for (int j=0;j<n;j++){
      int count = 0;
      for (int k=-1;k<=1;k++) count = count + A[(i-1)%n][(j+k)%n];
      count = count + A[i][(j-1)%n] + A[i][(j+1)%n];
      for (int k=-1;k<=1;k++) count = count + A[(i+1)%n][(j+k)%n];
      B[i][j] = count;
     }
```

NOTE: you should assume that `-1%n` equals `n-1` **[4]**

5. (a) What is a binary search tree? **[5]**

   (b) Does the order in which a fixed set of values is inserted into a binary search tree matter? Justify your answer. **[3]**

   (c) What property makes a binary search tree an AVL tree? **[2]**