The University of Nottingham

SCHOOL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

A LEVEL B MODULE, AUTUMN SEMESTER 2002-2003

ALGORITHMS AND DATA STRUCTURES

Time allowed TWO Hours

Candidates must NOT start writing their answers until told to do so

Answer QUESTION ONE and THREE other questions

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a dictionary to translate between that language and English provided that neither language is the subject of this examination. No electronic devices capable of storing and retrieving text may be used.

DO NOT turn your examination paper over until instructed to do so.

Additional Material:

1. Description of the Vector class

1. This multiple choice question is compulsory. In each part, select one answer. For parts (a) to (h), you get 3 points if you select the right answer and 0 if you don't. For part (i) you get one point if you select the right answer.

   (a) An algorithm's memory usage is described by the following function: $s(n) = 10n + 2nlog_2n$. What is the algorithm's space complexity (choose the tightest upper bound): (3)

   i. $O(1)$
   ii. $O(log\ n)$
   iii. $O(n)$
   iv. $O(n\ log\ n)$
   v. $O(n^2)$

   (b) Which one of the following is an invariant of the loop:

```
int indexGreatest = 0;
for (int i = 1; i < array.length; i++){
   if (array[i] > array[indexGreatest]) indexGreatest = i;
}
```
(3)

   i. for all `j`, if `0` $\leq$ `j` $\leq$ `i`, then `array[j]` $\leq$ `array[indexGreatest]`
   ii. for all `j`, if `0` $\leq$ `j` $<$ `i`, then `array[j]` $\leq$ `array[indexGreatest]`
   iii. for all `j`, if `0` $\leq$ `j` $<$ `i`, then `array[j]` $<$ `array[indexGreatest]`
   iv. for all `j`, if `0` $\leq$ `j` $\leq$ `i`, then `array[j]` $<$ `array[indexGreatest]`
   v. `indexGreatest = 0`

   (c) Suppose that the running time of an algorithm has a linear growth rate. On inputs of size 1000 it runs in 20 ms. What would be your estimate of its running time on inputs of size 10000? (3)

   i. 40 ms
   ii. 400 ms
   iii. 4000 ms
   iv. 200 ms
   v. 2000 ms

Question continued overleaf

(d) Which of the code fragments below has the following loop invariant: `result = n`$^{\mathtt{i}}$ (3)

i.
```
int power(int n, int k){
    int result = n;
    for (int i = 1; i < k; i++){
       result = result * n;
    }
}
```

ii.
```
int power(int n, int k){
    int result = 1;
    for (int i = 1; i < k; i++){
       result = result * n;
    }
}
```

iii.
```
int power(int n, int k){
    int result = n;
    for (int i = 1; i < k; i++){
       result = result * result;
    }
}
```

iv.
```
int power(int n, int k){
    int result = n;
    for (int i = 0; i < k; i++){
       result = result * n;
    }
}
```

v.
```
int power(int n, int k){
    int result = n;
    for (int i = 1; i < k; i++){
       result = n * n;
    }
}
```

Question continued overleaf

(e) What is the tightest upper bound on the growth rate of running time for the following algorithm: (3)

```
int algorithm(int n){
   int k = 0;
   while (n > 1) {
      n = n/2; k++;
   }
   return k;
}
```

i. $O(1)$
ii. $O(log\ n)$
iii. $O(n)$
iv. $O(n\ log\ n)$
v. $O(n^2)$

(f) Due to hash collisions, hash tables in Java use buckets to store multiple items with keys which hash to the same value. To find an item with a given key in the hash table, the key is hashed (which is a constant time operation), the right bucket accessed (in constant time) and then the bucket has to be searched sequentially. What is the worst case performance for search: (3)

i. constant time
ii. logarithmic in the number of buckets
iii. logarithmic in the number of items in the table
iv. linear in the number of buckets
v. linear in the number of items in the table

(g) Which data structure has reliably efficient (logarithmic) performance for search, insertion and deletion: (3)

i. unordered list
ii. ordered list
iii. ordered array
iv. binary search tree
v. balanced binary search tree

Question continued overleaf

(h) One of the sorting algorithms below works significantly faster if the input data is already sorted or almost sorted. Which one is it? (3)

i. bubble sort
ii. selection sort
iii. insertion sort
iv. merge sort
v. quick sort

(i) Binary search works on unordered arrays: (1)

i. yes
ii. no

2. (a) Using the following auxiliary `Node` class (15)

```
class Node {
   Object value;
   Node next;
   Node(Object v, Node n) {
      this.value = v;
      this.next = n; }}
```

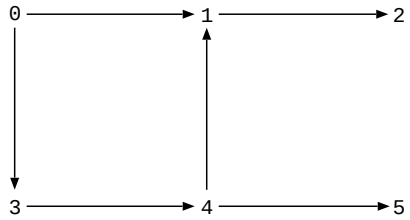
write a Java implementation of a simple single ended, single linked `List` class with the following methods:

- `public List() // constructor`
  Postcondition: creates an empty list
- `public void insert(Object o)`
  Postcondition: inserts `o` at the head of the list
- `public void remove()`
  Postcondition: removes the head of the list. If the list is empty, does nothing.
- `public boolean search(Object o)`
  Postcondition: returns true if `o` is in the list, false otherwise.
- `public Object head()`
  Postcondition: returns the object stored at the head of the list, without updating the list; if the list is empty, returns `null`.

(b) Using only the `List` class methods defined above, give an implementation of the following Stack ADT: (10)

- `public Stack() // constructor`
  Postcondition: creates an empty stack
- `public void push(Object o)`
  Postcondition: pushes `o` on top of the stack
- `public Object pop()`
  Postcondition: pops the stack and returns the value on top. If the stack is empty, returns `null`.
- `public Object peek()`
  Postcondition: returns the Object on top without updating the stack; if the stack is empty, returns `null`.

3. (a) How would you represent a directed graph as an adjacency matrix? Illustrate your answer using the following example: (5)



(b) Describe in English an algorithm to check whether a graph represented as an adjacency matrix contains a cycle (a path from some vertex to itself). Explain why this algorithm is correct. (5)

(c) Based on the algorithm you described above, give a Java implementation of the following method: (15)

```
boolean hasCycles(int[][] matrix, int n)
```

Precondition: `matrix` is a two-dimensional array containing only 0s and 1s, both dimensions are of size `n`.

Postcondition: the method returns `true` if `matrix` corresponds to a cyclic graph, and `false` otherwise.

You may change the contents of the `matrix`. Your solution may make use of additional methods which the `hasCycles()` method calls.

4. (a) What is a perfectly balanced binary tree? (3)

(b) Prove by induction that level i in a perfectly balanced binary tree contains 2^i nodes, assuming that the first level (the root) is level 0. (5)

(c) Prove by induction that a perfectly balanced binary tree with k levels has $2^k - 1$ nodes. (10)

(d) Give pseudocode for a method to search for an item given a key in a binary search tree. Show that in a perfectly balanced binary search tree, time complexity of search in the worst case is $O(log_2\ n)$, where n is the number of nodes in the tree. (7)

5. (a) What is a heap data structure? (5)

   (b) Describe how insertion and deletion in heaps works and why it makes them suitable for implementing priority queues. (5)

   (c) Write a Java class Heap which uses a Vector to store the contents of a heap. Vector class API description is enclosed. The class should contain a method to insert a new item in the heap (`void insert(Item i)` and a method which removes the root and returns it (`Item remove()`). Assume that items stored in the Heap are instances of the class (15)

```
class Item {
   Object value;
   int key;

   public Item(Object o, int i) {
      this.value = o;
      this.key = i;
   }

   public int key() {
      return key;
   }

   public Object value() {
      return value;
   }
}
```

   and the Heap is ordered on the `key()` values of the items.

6. Write a clear and well structured essay on the ways in which the differences between main memory and peripheral storage affect the design of algorithms and data structures. (25)