PRIFYSGOL CYMRU; UNIVERSITY OF WALES

DEGREE EXAMINATIONS MAY/JUNE 2003

SWANSEA

Computer Science

CS 121 Data Structures

Attempt 2 questions out of 3

Time allowed: 2 hours

Students are permitted to use the dictionaries provided by the University

Students are NOT permitted to use calculators

CS_121 Data Structures

(Attempt 2 questions out of 3)

Question 1.

(a) Build a binary search tree with the following ten items read from disk in the given order : London, Edinburgh, Southampton, Northampton, Eastham, Portsmouth, Swansea, Cardiff, Manchester, Liverpool. **[3 marks]**

(b) Given the definition of the *factorial* function :

fact(1) = 1
fact(n) = n * fact(n - 1)

write a recursive function to calculate fact(n). Give a call trace for computing fact(5). Write an iterative function for calculating factorials and comment upon the efficiency of each function. What type of data structures are suitable for recursion and for which type should recursion be avoided? **[10 marks]**

(c) Given the following type declarations :

```
nodePtr = ^node;
node = record
   data : integer;
   next : nodeptr
end;
```

complete the fragment of code for finding a node in a linked list. You may assume that the required record is *definitely* in the list (i.e. no need to check for a nil pointer), and that there is a variable called *head* which points to the first record in the list.

```
function find(wanted : integer):nodeptr;
begin
end;
```

[5 marks]

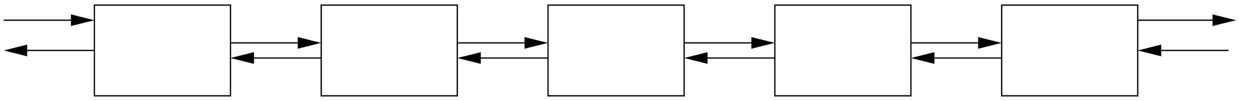
(d) What is meant by *covering* a hash table? What are the two methods we can use to ensure that a table is covered? Why do we need to ensure that a table is covered? When is a hash table considered to be full? **[7 marks]**

Question 2.

(a) Give (using the big-O notation) the complexity for searching in a binary search tree, an unordered linked list and a hash table. Explain in each case how the performance is achieved.

[6 marks]

(b) Given the following two-way linked list :



and the following data type declarations :

```
nodePtr = ^node;
node = record
   prev, next : nodeptr;
   data : integer
end;
```

complete the following code fragment which sets the appropriate pointers and deletes the node to which the pointer *this* is pointing :

```
procedure delete(data : integer);
var this : nodeptr;
begin
   //Code to set this to the required node
end;
```

[5 marks]

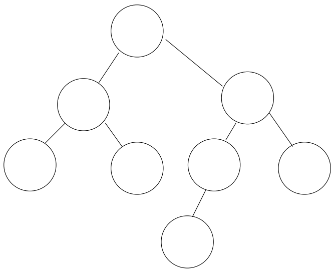
(c) The table below gives a list of names in the order in which they are read from disk. The next two columns give a primary(p) and secondary(s) hash function for each name:

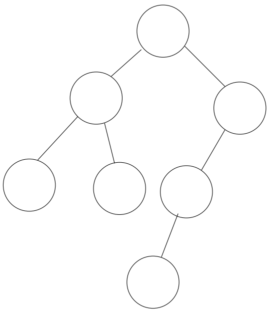
Name	p	s
Morris	2	2
Pauline	5	2
Yannis	2	3
Nicholas	1	1
Robin	6	1
Lucy	5	2

Insert these names into a hash table containing 7 entries indexed 0 to 6 in the order given using seperate chaining, linear probing and secondary hashing. (Probing sequences should be incremented *not* decremented). **[9 marks]**

Continued overleaf

(d) Give the balance factor at each node in the following two trees and state whether each tree is, or is not, an AVL Tree :





[5 marks]

Question 3.

(a) Explain the problem which occurs when deleting an entry from a hash table. How is this problem overcome? **[3 marks]**

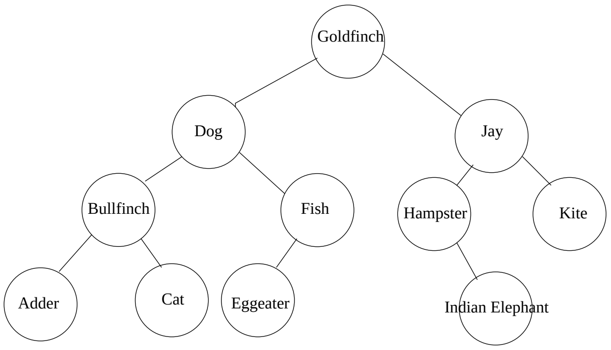
(b) How is the *load factor* of a hash table defined? What does the load factor influence?

[4 marks]

(c) A *queue* is described as a *LILO* (Last In Last Out) data structure. Name and describe 5 operations which might be carried out upon such a structure. Give *exact specifications* for these operations. **[5 marks]**

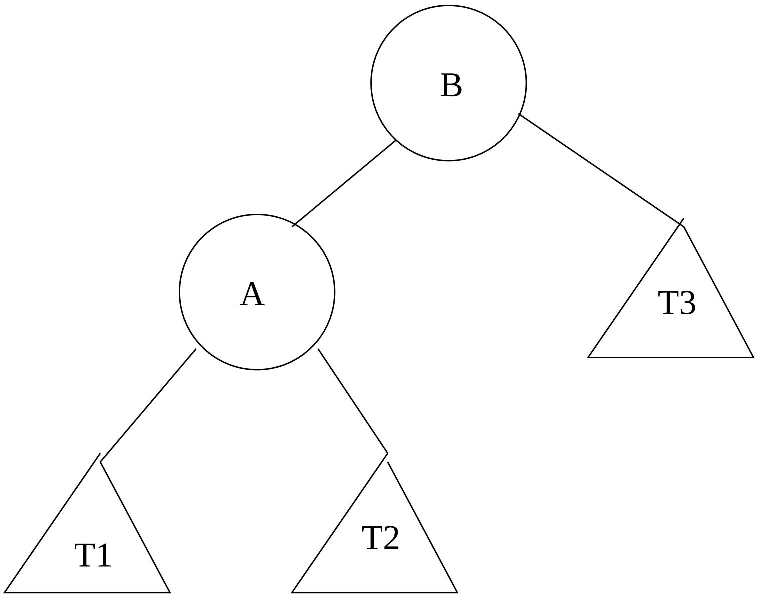
(d) Describe two different methods by which memory can be allocated at run time. What problems can occur and how are they overcome? What data structure should you use to keep track of spare memory blocks. **[4 marks]**

(e) There are three different cases for deleting a node from a binary search tree. These can be demonstrated in the following tree by the nodes Cat, Fish and Goldfinch. Describe each case and show the resulting tree after each deletion. (Start with the original tree in each case).



[6 marks]

(f) In the following tree A and B are nodes while T_1, T_2 and T_3 are subtrees. Draw the tree after a single right rotation about the root of the tree.



[3 marks]