HCI Computing 2011 Prelim Paper 2 Solution Guide

1.	(a)	• Ordered • Sorted based on Key field • Suitable for high hit rate applications • Ideal for master files	[2]
	(b)	Delete algorithm • Transfer records to another file till matching key value • Skip to the record and transfer till the EOF	[4]
	(c)	i. • Allows direct access to record • Unordered • Loan records are place based on a hash function • Using a key field such as Student ID	[4]
		ii. • If the hash function allocates a record to an occupied location • Place in the next available location • If EOF, search from top of file to search for empty slot • Till the initial allocated address H(k)	[4]
		iii. • Low hit rate (Only access when a user returns /borrow books) • Fast, allow direct access to specific record rather than from previous records in table	[2]

2.	(a)	Cycle refer to the number of instructions that can be executed by the processor in 1 second. Each instruction goes through the fetch-execute-cycle. Fetch-Decode-Execute-Store	[5]
	(b)	• Memory access time is the amount of time between a memory operation request (read or write) and the time the memory operation completes. • Modern CPUs are so much faster than memory • The slower the memory access speed the more time it will take to transfer data between in/out of memory.	[2]
	(c)	i. • If the differing speed is not synchronized it will result unsuccessful operations or garbage being sent • Not optimizing the use of the processor	[1]
		ii. Spooling • Placing data/instructions in temp storage while waiting for execution • Removed from storage once processed	[3]

3. (a) Data consistency

- 2 records for Clarence & Smith
  - 2 different date joined for Smith or
  - 2 different address for Clarence

Data integrity

- 2 records for Clarence & Smith
  - Which to accept as the official data

[4]

(b) Suggested answers:

Teachers(TID, Name, Address, DOB, Type, Date Joined)
Students(SID, Name, Address, DOB, Type, Date Joined)
Access(A_ID,ID,Lvl)

[6]

(c)

Teacher

Name	Address	Date_Of_Birth	Mobile	ID	Type	Date Joined
Hon Yew Peng	67 Pasir Ris	09-Sep-67	97654534	12	Principal	01/03/1988
Tan Hock Joo	19 Steven Rd	03-Mar-65	67654321	34	Teacher	02/03/1999
Smith Jones	212 Holland Road	01-Jan-77	97865431	100	Teacher	01/03/2007

Student

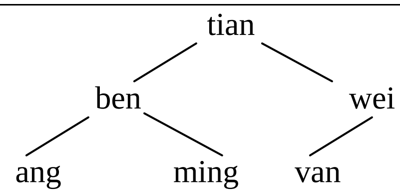
Name	Address	Date_Of_Birth	Mobile	ID	Type	Date Joined
Janice Tan	56 Clementi Road	12-Dec-95	97896541	1345	Student	01/01/2010
Clarence Teo	122 Ang Mo Kio	04-Mar-96	98653431	3245	Student	02/01/2011

Access

A_ID	ID	LVL
A1	12	5
A2	34	3
A3	100	4
A4	1345	1
A5	3245	2

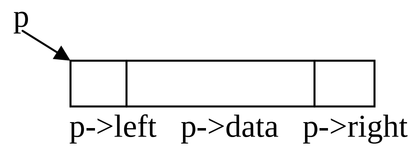
[3]

4. (a)



[2]

(b) The algorithm is best expressed recursively in terms of the notation given below:



Root is a pointer points to the root node of the binary search tree

```
ptr  =   Root                          // initialization
TreeInsert( ptr, item)                 // ptr points to the root node of the tree (or subtree)
      if  ( ptr = null )               //  means can insert here
            ptr = new tree node        //  create a new node and set ptr to point to it
            ptr->data = item
            ptr->left = null
            ptr->right = null
            if ( Root = null ) then  Root  = ptr       //  empty tree, i.e. insert as root
     else
            if  ( item < ptr->data )  then             //  try to insert to the left of current node
                  TreeInsert( ptr->left, item)
            else
                  TreeInsert( ptr->right, item)      //  try to insert to the right of current node
            endif
    endif
endOfTreeInsert
```

[5]

(c)

i. In-order : ang, ben, ming, tian, van, wei [2]
ii. Pre-order: tian, ben, ang, ming, wei, van [2]
iii. Post-order: ang, ming, ben, van, wei, tian [2]

(d)

		Left Pointer	Data	Right Pointer
Root	1	2	tian	4
	2	5	ben	3
	3	-1	ming	-1
	4	6	wei	-1
	5	-1	ang	-1
	6	-1	van	-1
Free	7	8		
	8	9		
	.	.		
	.	.		

Three 1-dimensional arrays: Left Pointer, Data and Right Pointer

Left Pointer to point to the left child of a node
Data to store the content of the node
Right Pointer to point to the right child of a node

The value in the left and right pointer array indicates the index of the array where the node is pointing to. A value of -1 indicates that there is no child node.

There is also a root pointer keeps a record of the subscript of the root of the tree, in this case 1, and the head of the list of free spaces, in this case 7. Free space is linked through the left pointer array. [5]

(e) Insert(x, item) – inserts a new value, item, into the binary search tree x. [1]
Delete(x, item) – deletes the value, item, from the binary search tree x. [1]
Search(x, item) – returns true if the value, item, is in the binary search tree x, otherwise returns false.
Size(x) – returns the number of nodes in the binary search tree x.
IsEmptyTree(x) – returns true if the binary search tree x is empty, otherwise returns false.
Create(x) -- creates an empty binary search tree x.
SortedDisplay(x) -- displays values stored in the binary search tree x in sorted order.

---

5. (a) 2-D array of car records (10 by 20)
Each car record has a data structure of three fields as shown: colour, door, sunroof

(b) (i)

```
found = false
Set row and col to 1    // initialize to lowest
Repeat
        Repeat
                If shelf[row,col] is free then
                        found = true
                Else
                        col = col + 1
                Endif
        Until (found or col > 20) // all cols in a particular row is looked at
        row = row + 1
Until (found or (row > 10) // all rows looked at
If found, output “Free slot at, “[row, col]
```

(ii)

```
found = false
Set row to 10 and col to 20  // initialize to highest
Repeat
        Repeat
                If car at shelf[row, col] is of required type then
                        found = true
                Else
                        col = col - 1
                Endif
        Until (found or col < 1)
```

```
        row = row - 1
Until (found or row < 1) // all shelves examined
```

(iii)

```
Data structure to store quantity: Qty[5,2,2]
Set qty data structure to 0
Set row and col to 1
Repeat
        Repeat
                If shelf[row,col] is occupied then
                        qty[colour, door, sunroof] = qty[colour, door, sunroof] + 1
                Endif
                col = col + 1
        Until col > 20
        row = row + 1
Until row > 10
Write out headings “Colour    Doors    Sunroof     Qty”
Loop for all values of colour, doors, sunroof
        Write out colour, doors, sunroof and qty [colour, doors, sunroof]
Endloop
```

[16]

(c) (i)

```
<code> :== <letter><NZD> | <letter><NZD><digit> | <letters>
<NZD> :== 1|2|3|4|5|6|7|8|9
<letters> :== <letter> | <letter><letters>
```

(ii) Suitable syntax diagram [6]

6. (a) Repeat… until (post-test)
While (pre-test)
For (step-wise increment/decrement) [3]

(b) A function which contains a call to itself.
Used when the original task can be reduced to a simpler version of itself
Should include at least one terminal case – a case that contains no further calls to the recursive subprogram so that it will not continue indefinitely. [3]

(c) 2 advantages:
- When solution to a problem is essentially recursive (such as tree traversal), enables programmer to write a program which mirrors the solution.
- Recursive solutions are often much shorter than non-recursive ones.

2 disadvantages
- If the recursion continues too long, the stack of return addresses may become full (ie no available memory is left) and the program will crash.
- Recursive routines can be difficult to follow and to debug. [4]

(d) Return address + values of the parameters (state of the function at that point in time) [2]

(e) Stack – FILO data structure [2]

(f)

35

5 * 3 + Magic(4) 20

4 * 5/2 + Magic(3) 10

3 * 4/2 + Magic(2) 4

2 * 3/2 + Magic(1)

1

[4]