Solutions to Summer 09 midterm :
(The numbers in square bracket denote the maximum scores.)

1. [1] point each
(a) T
(b) T
(c) F
(d) F
(e) T
(f) T
(g) T
(h) F (exception: root)
(i) F
(j) F (although the real sizes are same, estimate may differ)

2.
[3] (a) ceil((10000*10+10) / 5000) = 21

Explanation: We need to store search key per block (10 bytes per block), and one block pointer (10 bytes) that points to the first block. Block pointers per block are not required since the blocks are contiguous, and so a block pointer can be computed using the offset from the first block pointer.

[3] (b) 10000 / (5000 / (10+10)) = 40

[4] (c) ceil(10000 * 10 / floor(5000 / (10+20))) = 603

Note: Taking floor as above is necessary to make sure that records do not span blocks. However, for now we gave full credit to students who assumed that records span blocks, and computed number of blocks as
10000 * 10 / (5000 / (10+20)) = 600.

3.
[4] (a)
00: (0000,0100,1000)
01: (0111,0001,1111,0111) --- (0011,0011,0001)
10: (1110, 0010)
[4] (b)
000: (0000,1000)
001: (0001,0001)
010: (0010)
011: (0011,0011)
100: (0100)
101: ()
110: (1110)
111: (0111,0111,1111)
[2] (c) 12/32 = 3/8

4.
[3] (a) 100/10 = 10
[3] (b) 300/5 = 60
[4] (c) 100*50/5 + 300*100/5 + 200*200/10 = 11000

5.
[3] (a) 1500*4 + 1500 + 800 = 8300
[3] (b) 800 + 20000*k

[1] (c) 800 + 20000*k < 8300 <=> k < 7500/20000 = 3/8
[3] (d) 800 + 20000*45000 = 900000800

Note: For part (a), only 30 buffers were provided in the original exam, which made the question difficult. We corrected it to 50 buffers for students taking exam in class. Those remote students who did not receive this correction were given full credit if they displayed understanding of the issues involved with 30 buffers. Note that this typo did not affect part b, c or d of the question.

6.
[3] (a) (n+1)^j*n - (n+1)^(j-1)*n = n^2(n+1)^(j-1)

Explanation: In full tree, each node contains n+1 node pointers. ((n+1)^(j-1))*n is the number of records indexed by tree of height j. So, (n+1)^j*n - (n+1)^(j-1)*n is the number of records to be added.

[3] (b) ceil(1+log_{n+1}(r/n))

Explanation: For minimum height, each node will have n+1 pointers. r = (n+1)^(j-1)*n

[4] (c) floor(2+log_{ceil((n+1)/2)}(r/(2*floor((n+1)/2))))

Explanation: For maximum height, each node will have atleast p=ceil((n+1)/2) pointers with the exception of root which can have as low as 2 pointers. Leaf nodes point to l=floor((n+1)/2) records. Therefore, r = 2*(p^(j-2))*l