A27217 *No Calculator permitted in this examination*

UNIVERSITY OF BIRMINGHAM

School of Computer Science

First Year – Degree of BSc with Honours

Artificial Intelligence and Computer Science

Computer Science

Computer Science with Study Abroad

Computer Science with Business Management

Natural Sciences

First Year – Degree of BEng/MEng with Honours

Computer Science/Software Engineering

First Year – Joint Degree of BSc/MSci with Honours

Mathematics and Computer Science

06 22754

Foundations of Computer Science

Main Examinations 2010

Time allowed: 3 hours

[Answer ALL Questions]

Question 1 [Tools for measuring program efficiency and achieving program correctness]

(Total marks for this question 20%)

(a) Calculate the upper and lower logarithms in base 2 of the following numbers: [5%]

(i) 127.

(ii) 128.

(iii) 1100.

(iv) 1024 * 256 * 512. Show work.

(b) Write a method (in Java or Groovy) for computing and returning the lower logarithm of a given positive integer using a while-loop. [5%]

(c) Provide an invariant for the loop you produced in the previous item which (i) holds before the loop starts, (ii) holds every time the loop body ends, and (iii) when the loop ends, the negation of the loop condition together with the invariant imply that the value returned by the algorithm is correct. [5%]

(d) Justify that your answer to the previous item does achieve the required goals (i)–(iii). [5%]

Question 2 [Algorithm understanding, construction and justification]
(Total marks for this question 30%)

The following is a binary search algorithm that finds the first position a given element x is or could be in a given sorted array a:

```
int firstPosition(int x, int [] a) {
  // Pre-condition: the array a is in ascending order.
  //
  // Specification:
  //
  // We calculate and return the number s such that:
  //
  //    a[i] <  x for all indices i <  s, and
  //    a[i] >= x for all indices i >= s.

  int lower = 0;
  int upper = a.length;

  while (lower != upper)
    // Invariant: 0 <= lower <= s <= upper <= a.length
  {
    int mid = (lower+upper)/2;
    assert(lower <= mid  &&  mid < upper);
    if (x <= a[mid])          // s <= mid
       upper = mid;
    else                      // mid < s
       lower = mid+1;
  }

  return(lower);

  // Justification of correctness:
  //
  // Because lower == upper is the negation of the while-loop condition,
  // we conclude that lower == s == upper, by the invariant.
}
```

(a) If $a = \{4, 4, 5, 5, 6, 7, 8, 9, 9, 9\}$, what will the algorithm return for the following choices of x? [5%]

(i) $x = 3$.

(ii) $x = 4$.

(iii) $x = 5$.

(iv) $x = 9$.

(v) $x = 11$.

(b) How many times is the loop body executed as a function of the length of the given array? [5%]

(c) Justify your answer to the previous question. [5%]

(d) Modify the algorithm so that it instead finds the *last* position of `x` in the array `a`, making sure you

(i) obtain an algorithm as efficient as the above, [5%]

(ii) provide a correct specification, [5%]

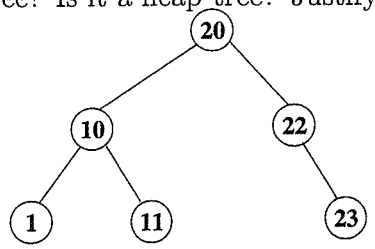
(iii) provide correct invariant, assertions, and comments, and give a valid justification of correctness. [5%]

[PART B. USE NEW ANSWER BOOK.]

Question 3 [TREES]

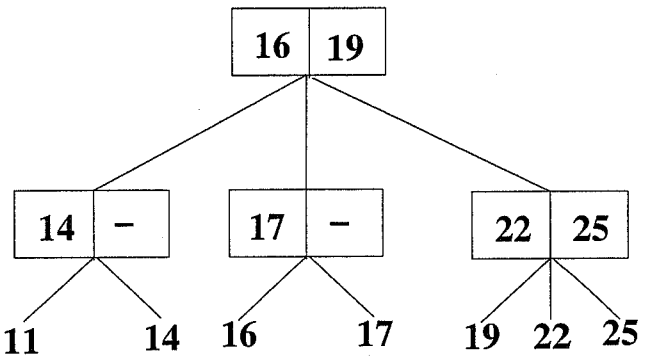
(Total marks for this question 17%)

(a) Define a binary search tree and define a heap tree. Is the following tree a binary search tree? Is it a heap tree? Justify your answers.



[4%]

(b) Insert the numbers 12 and 15 in this order in the 2-3 tree below. Display the resulting 2-3 tree. Would the resulting 2-3 tree look different if you inserted first 15 and then 12? Explain your answer in detail. [5%]



(c) Discuss the relative merits of linked lists, binary search trees, 2-3 trees, and hashtables with respect to finding, inserting, and deleting of elements. Discuss in particular the time complexity of each of the operations. (Make favourable assumptions about the hashtable. What are such favourable assumptions?) [4%]

(d) Write a procedure in pseudocode which computes the average value of the keys represented in a heap tree. What is the complexity of your algorithm? Can that be improved to a better complexity class?

[4%]

Question 4 [SORTING]

(Total marks for this question 16%)

(a) Describe selection sort and sort the array $[7, 4, 8, 2, 9, 1]$ with it (increasing order), showing each intermediate array in which a new element is selected. For each intermediate array indicate which part of the array is sorted. Also indicate how many comparisons and how many variable assignments are necessary in total for each intermediate array. [4%]

(b) Assume you have to sort an array with $n = 1{,}000{,}000$ elements. Compare the complexities of insertion sort and heapsort. How long would each algorithm roughly need assuming each basic step takes one millisecond. [4%]

(c) Sort the following array `a` using quicksort,

$$[6, 11, 4, 9, 8, 2, 5, 8, 13, 7]$$

The pivot should be chosen as the arithmetic mean of the first and the last element, i.e., $(\texttt{a[0]} + \texttt{a[size} - 1])/2$ (rounded down).

Show all important steps such as partitioning and the recursive calls to the algorithm. [4%]

(d) A sorting algorithm is called *stable* if it preserves the relative order of any two elements with equal keys. Under which conditions is quicksort stable? Explain your answer in detail. [4%]

Question 5 [GRAPHS]

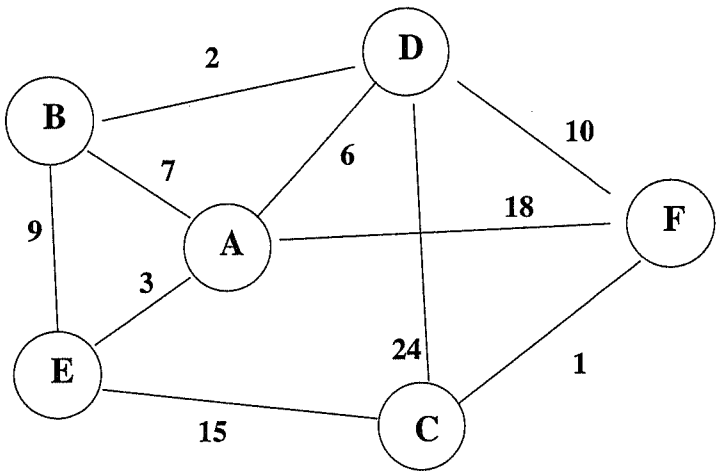
(Total marks for this question 17%)

(a) Display the graph corresponding to the following weight matrix (representing asymmetric distances).

	A	B	C	D	E
A	0	2	∞	22	33
B	∞	0	14	24	5
C	∞	28	0	∞	33
D	22	16	∞	0	∞
E	9	∞	∞	∞	0

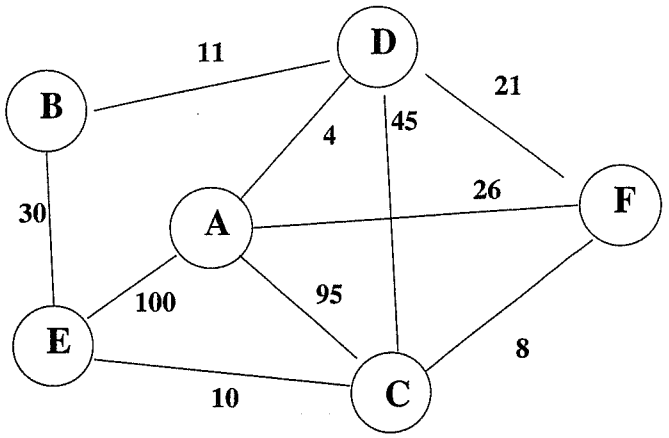
[4%]

(b) Let the following weighted graph be given:



Apply Prim's algorithm to determine a minimal spanning tree. Display the graph after each step. [4%]

(c) Let the following weighted graph be given:



Use Dijkstra's algorithm to find the shortest path from **A** to **E**. What is the minimal cost? Show all relevant steps in detail. [5%]

(d) Outline a proof that Prim's algorithm constructs a minimal spanning tree. Make use of an invariant. Why does Prim's algorithm terminate? [4%]

End of Paper