UNIVERSITY OF SOUTHAMPTON COMP1201W1

SEMESTER 2 EXAMINATION 2013/2014

ALGORITHMICS

Duration: 120 mins

You must enter your Student ID and your ISS login ID (as a cross-check) on this page. You must not write your name anywhere on the paper.

Student ID:	
ISS ID:	

Question	Marks
1	
2	
3	
4	
Total	

Answer all parts of the question in section A (25 marks) and TWO questions from section B (30 marks each).

This examination is worth 85%. The tutorials were worth 15%.

University approved calculators MAY be used.

A foreign language translation dictionary (paper version) is permitted provided it contains no notes, additions or annotations.

Each answer must be completely contained within the box under the corresponding question. No credit will be given for answers presented elsewhere.

You are advised to write using a soft pencil so that you may readily correct mistakes with an eraser.

You may use a blue book for scratch—it will be discarded without being looked at.

Number of pages 16

Section A

Question A 1

(a) Write pseudo code for the quick sort algorithm. What is its average and worst case time complexity? *(5 marks)*

The quicksort algorithm is as follows

```
QuickSort($a$, left, right) {
  if (right-left < threshold)
    InsertionSort($a$, left, right)
  else
    pivot = ChoosePivot($a$, left, right)
    part = Partition($a$, pivot, left, right)
    QuickSort($a$, left, part)
    QuickSort($a$, part+1, right)
  endif
}
```

$\overline{5}$

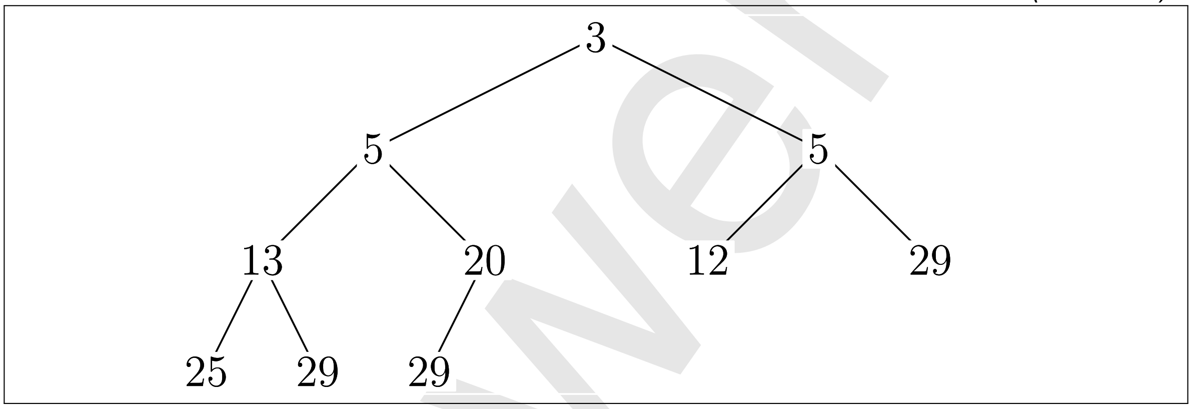
Average case complexity: $\Theta(n \log(n))$

Worst case complexity: $\Theta(n^2)$

(b) Heaps use a binary tree encoded into an array. Show the binary tree represented by the following array.

3	5	5	13	20	12	29	25	29	29

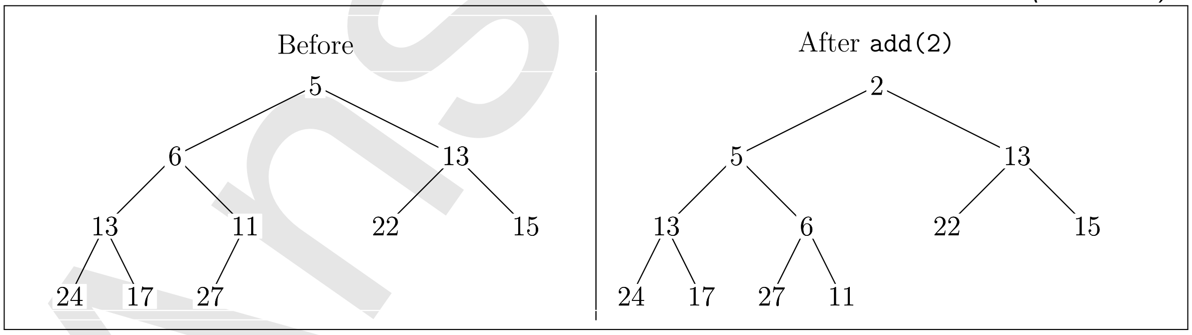
(1 marks)



$\overline{}$ 1

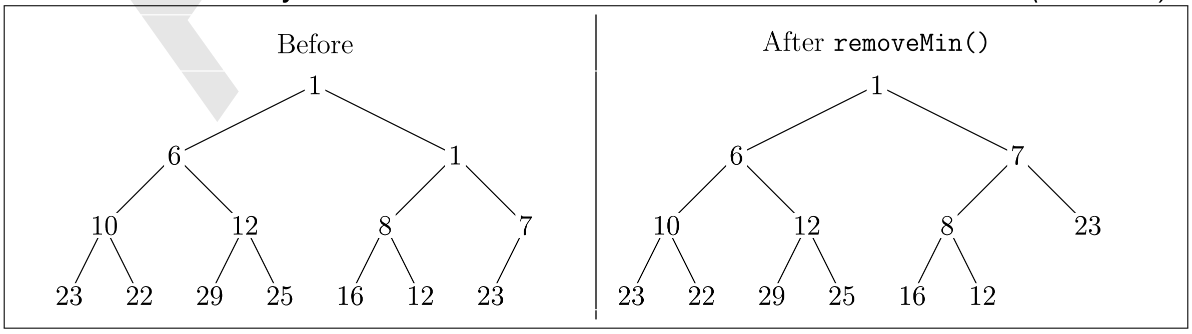
(c) Show what happens to the heap shown on the left when you add 2.

(2 marks)



$\overline{}$ 2

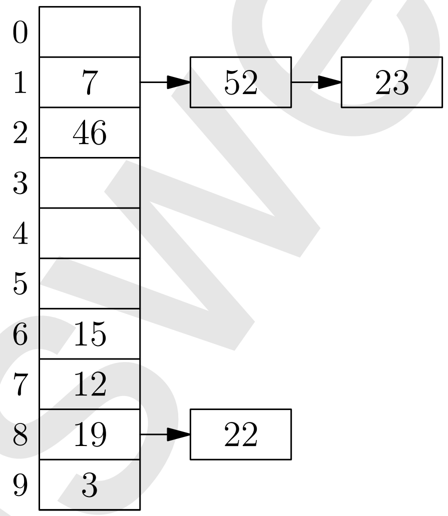
(d) Show what happens to the heap shown on the left when you remove the minimum entry. *(2 marks)*



$\overline{}$ 2

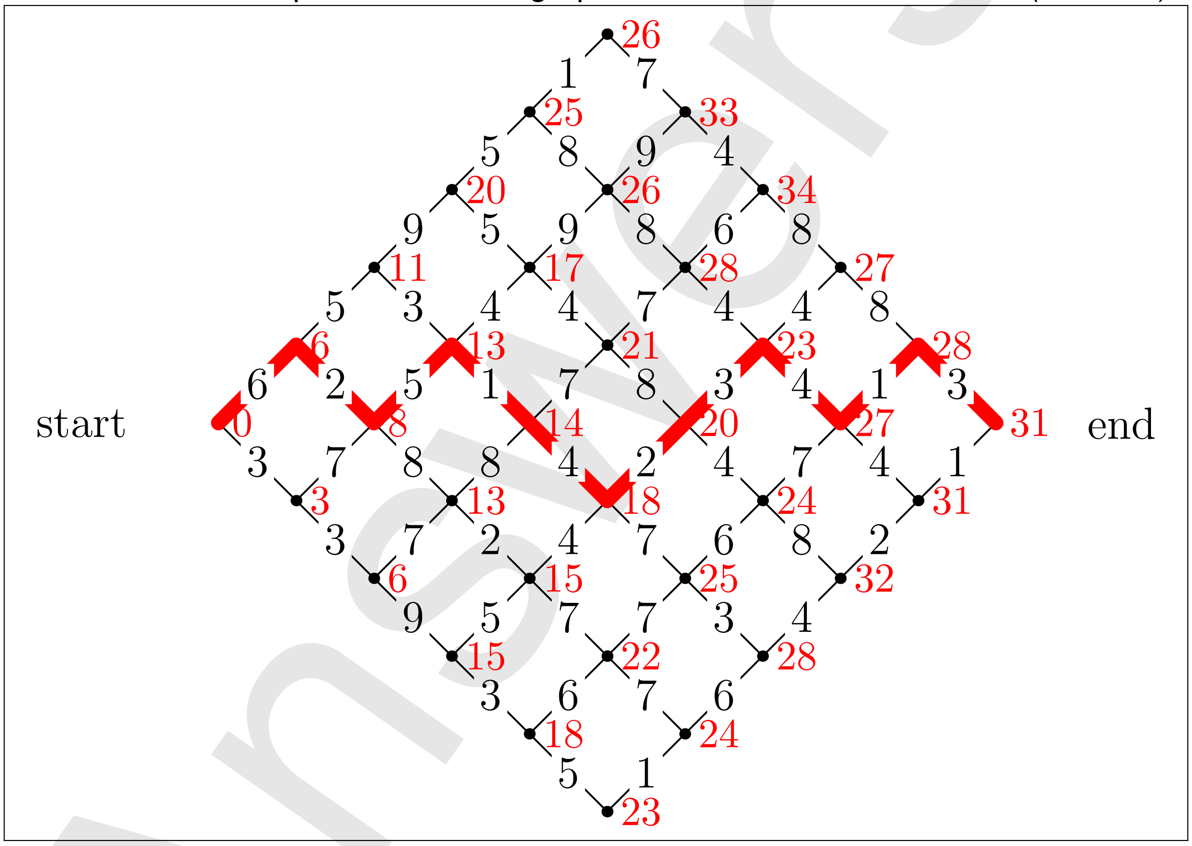
(e) Show how the numbers 12, 3, 7, 19, 15, 52, 46, 23, 22 would be hashed using a hash function $d_2 + 3d_1$ where d_1 is the first (least significant) digit and d_2 the second digit. Show how these would be stored in a hash table using separate chaining. *(10 marks)*

d_2d_1	12	3	7	19	15	52	46	23	22
$d_2 + 3d_1$	7	9	21	28	16	11	22	11	8
$(d_2 + 3d_1)\%10$	7	9	1	8	6	1	2	1	8



$\overline{10}$

(f) Use the dynamic programming forward algorithm to compute the minimum cost of each path from the left most node to each other node where the cost of moving along an edge is equal to the number shown. An edge can only be traversed from left to right. Use the backwards algorithm to find the minimum cost path across the graph. *(5 marks)*



$\overline{}$
5

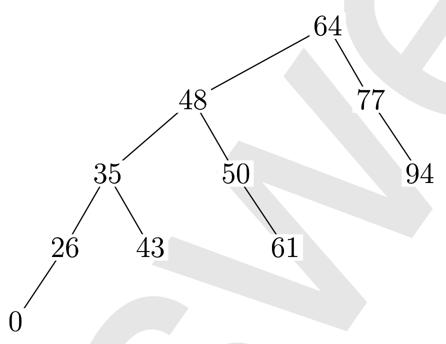
End of question 1

Q1: (a) $\frac{}{5}$ (b) $\frac{}{1}$ (c) $\frac{}{2}$ (d) $\frac{}{2}$ (e) $\frac{}{10}$ (f) $\frac{}{5}$ Total $\frac{}{25}$

Section B

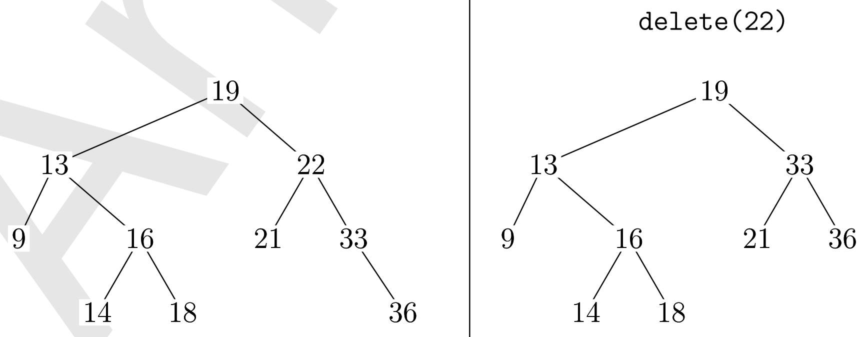
Question B 2

(a) Draw the binary search tree produced when inserting 64, 48, 35, 50, 26, 77, 43, 94, 0, 61. *(2 marks)*



$\frac{}{2}$

(b) Draw the tree obtained by deleting 22 from the binary search tree shown. *(2 marks)*



$\frac{}{2}$

(c) Describe the two rules that define an AVL tree *(4 marks)*

(i) The heights of the left and right subtree differ by at most 1

(ii) The left and right subtrees are AVL trees

(Although the rules are easy to express this requires quite detailed knowledge.)

(d) Let $m(h)$ be the minimum number of elements in an AVL tree of height h. Write down a recurrence relationship for $m(h)$ *(4 marks)*

$\boxed{\overline{4}}$

(Tests understanding of recurrence, but in a setting they may not expect.)

Since the left and right subtrees cannot differ by more than 1.

$$m(h) = m(h-1) + m(h-2) + 1$$

$\boxed{\overline{4}}$

(e) Write down the boundary condition for $m(1)$ and $m(2)$ *(2 marks)*

Clearly the minimum height of any tree binary with 1 or two elements are

$$m(1) = 1 \qquad\qquad m(2) = 2.$$

$\boxed{\overline{2}}$

TURN OVER

(f) Using proof by induction show that the minimum number of elements in an AVL tree of height h is greater than or equal to $b(h) = \left(\frac{3}{2}\right)^{h-1}$ *(6 marks)*

We first verify the base cases

$$m(1) = 1 \geq b(1) = \left(\tfrac{2}{3}\right)^0 = 1$$
$$m(2)2 \geq b(2) = \left(\tfrac{2}{3}\right)^1 = \tfrac{2}{3}$$

To prove the induction case we start from the recurrence relation and substitute in the bound

$$\begin{aligned} m(h) &= m(h-1) + m(h-2) + 1 \\ &\geq b(h-1) + b(h-2) + 1 = \left(\tfrac{3}{2}\right)^{h-2} + \left(\tfrac{3}{2}\right)^{h-3} + 1 \\ &> \left(\tfrac{3}{2}\right)^{h-2} + \left(\tfrac{3}{2}\right)^{h-3} = \left(\tfrac{3}{2}\right)^{h-3}\left(\tfrac{3}{2}+1\right) = \left(\tfrac{3}{2}\right)^{h-3}\tfrac{5}{2} \\ &> \left(\tfrac{3}{2}\right)^{h-3}\tfrac{9}{4} = \left(\tfrac{3}{2}\right)^{h-1} \end{aligned}$$

$\boxed{\overline{6}}$

(g) Use this bound to show that the complexity of insertion and search for an AVL tree is $O(\log(n))$. *(3 marks)*

Insertion and search just depends on the depth of the tree. Since the minimum number of elements in an AVL tree of depth h is at least $\left(\frac{3}{2}\right)^{h-1}$. A tree of depth n will have a height no more than

$$\frac{\log(n)}{\log(3/2)} + 1 = O(\log(n)).$$

$\boxed{\overline{3}}$

This is then a bound on the depth.

(h) What is the average and worst case time complexity of search in an unbalanced binary search tree and why is balancing a binary search tree regarded as so important? *(3 marks)*

Average case complexity: $\Theta(\log(n))$

Worst case complexity: $\Theta(n)$

The worst case however happens when the elements are inserted in order (or in reverse order). This is not an uncommon occurrence so to make the data structure robust it is important to makes sure the trees are balanced.

$\boxed{\overline{}\,3}$

(i) Describe how a tree map is implemented in the Java collections. *(4 marks)*

A tree map is implemented using a tree set where the elements of the set are of type `Entry<K,V>` which is a container

```
class public Entry<K,V> implements Comparable {
  K key;
  V value;
}
```

The tree map implements the Map interface (it does not extend the tree set).

$\boxed{\overline{}\,4}$

End of question 2

Q2: (a) $\frac{}{2}$ (b) $\frac{}{2}$ (c) $\frac{}{4}$ (d) $\frac{}{4}$ (e) $\frac{}{2}$ (f) $\frac{}{6}$ (g) $\frac{}{3}$ (h) $\frac{}{3}$ (i) $\frac{}{4}$ Total $\frac{}{30}$

Question B 3

(a) Describe the difference between depth first search and breadth first search. Give an application of each. *(4 marks)*

Depth first search explores the most recently found node, while breadth first search explores the nodes in the order in which they are discovered.

$\overline{4}$

DFS Application: Finding articulation (biconnectivity, etc.)

BFS Application Finding sortest path between two nodes

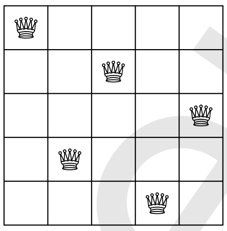
(b) What is topological sort? Give an application. *(4 marks)*

Given a partial ordering defined by a directed acyclical graph (DAG), a topological sort is one ordering which respects the partial ordering (i.e. for any pairs of node A and B connected by an edge (A, B), the ordering must have A occurring before B).

$\overline{4}$

Application: find an order in which to perform a set of dependent tasks for constructing some product.

(c) The n-queens problem is to put n queens on a chess board such that no queen is in the same row, column or diagonal as any other queen. Either write pseudo code or describe in outline an algorithm to solve the n-queens problem.



(10 marks)

(Any approximation to the following code is sufficient)

```
List nextRow(row, positionList) {
  if (row==n) {
     return positionList
  }
  for col = 1 to n {
    if legalQueen(col, row, positionList) {
       positionList.add(col)
       solution = nextRow(row+1, positionList)
       if (solution ≠ null) {
         return solution
       }
    }
  }
}
```

$\overline{}$
10

where `legalQueen(col, row, positionList)` is true if placing a queen in column `col` of row `row` does not interfere with any of the queens in the earlier rows given by `positionList`.

TURN OVER

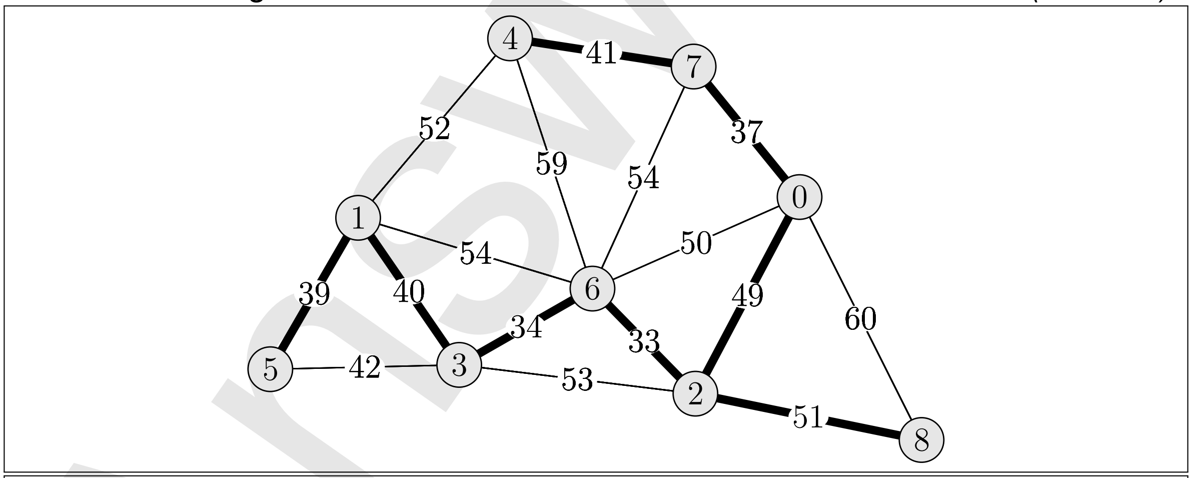
(d) Describe Kruskal's algorithm for constructing a minimum spanning tree. What data structures are need to implement the algorithm? *(6 marks)*

In Kruskal's algorithm we consider each edge in order and add it to the minimum spanning tree provided that in adding the edge we do not form a cycle.

To order the edges we can use a priority queue, although we can use any sorting algorithm we like. To determine whether an edge forms a cycle we the disjoint set (aka union-find) data structure.

$\overline{6}$

(e) In the graph below draw the minimum spanning tree (by highlighting the edges) and write down the order in which the edges would be found by Kruskal's algorithm. *(6 marks)*



1.	$(2, 6)$	**2.**	$(3, 6)$
3.	$(0, 7)$	**4.**	$(1, 5)$
5.	$(1, 3)$	**6.**	$(4, 7)$
7.	$(0, 2)$	**8.**	$(2, 8)$

$\overline{6}$

End of question 3

Q3: (a) $\frac{}{4}$ (b) $\frac{}{4}$ (c) $\frac{}{10}$ (d) $\frac{}{6}$ (e) $\frac{}{6}$ Total $\frac{}{30}$

Question B 4

(a) Describe the local (neighbourhood) search strategy. What is its main drawback? *(4 marks)*

Local search explores the search space by taking a potential solution and modifying it. It accepts the modification if the new solution is better (lower cost) than the current solution, otherwise it stays put.

Drawback: The major drawback is that it can get stuck in local optima.

$\boxed{\overline{\ \ }\atop 4}$

(b) Describe a strategy to overcome this. *(4 marks)*

(I would expect either simulated annealing or genetic algorithms—or even tabu search.)

Simulated annealing is a modification that allows the searcher to make a non-improving move with some probability depending on how bad the move is. That probability decreases over time. This allows the search to escape local optima, particularly early on. Because there is a bias towards good solutions, the searcher is likely to spend most of its time around higher quality solutions.

$\boxed{\overline{\ \ }\atop 4}$

(c) How efficient would you expect heuristic search to be on a problem like TSP or graph colouring? *(2 marks)*

These are NP-hard problems where we would expect any search to take super-polynomial time to have a good chance of finding the optimal solution for many instances.

$\boxed{\overline{}\,2}$

(d) Describe the three conditions required of a linear programing problem. *(6 marks)*

(i) The objective function must be linear

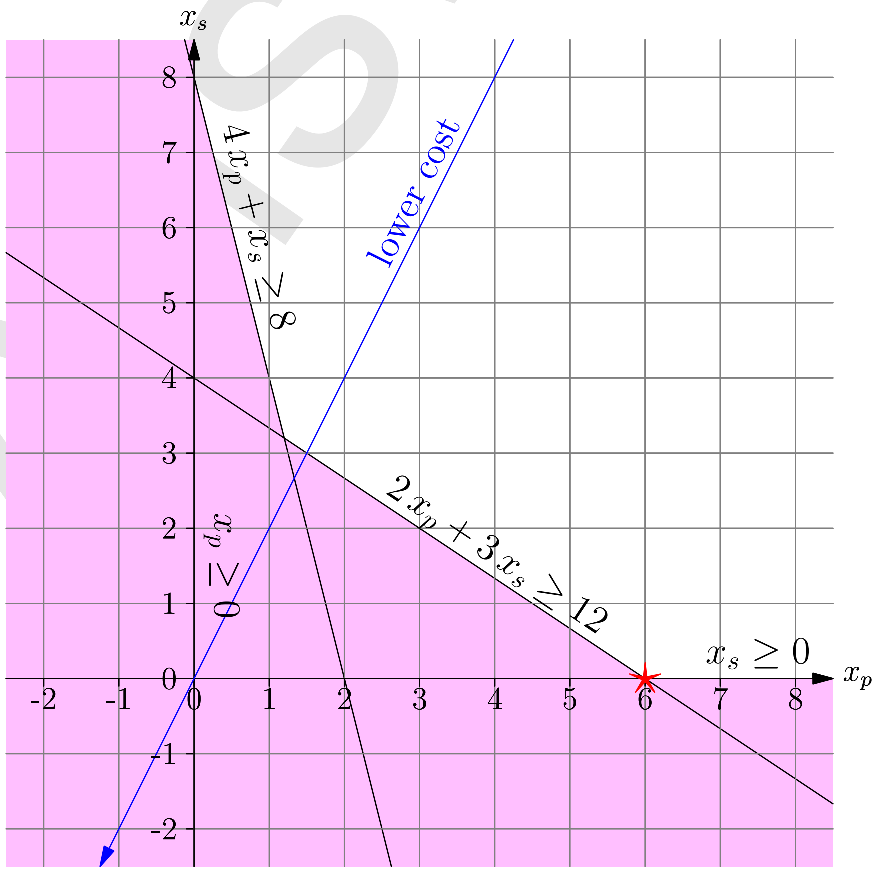
(ii) The constraints must be linear

(iii) All the quantities must be non-negative

$\boxed{\overline{}\,6}$

(e) Consider the problem of deciding the cheapest balance between potatoes and soybean. Let x_p be the amount of potatoes and x_s the amount of soybeans. The cost of soybean is taken to be twice the cost of potatoes. We require the vitamin C content from the two ingredients should be at least 12 units where the vitamin content of 1 unit of potatoes is 2 and for soybean 3. We also require the vitamin B6 content of the two ingredients should be at least 8 units where one unit of potatoes supplies 4 while soybean provides 1 unit. Write down the linear programming problem and draw the direction that minimises the cost and the constraints (shade the infeasible region). Mark the optimal solution. *(10 marks)*

minimise	$x_p + 2\,x_s$
subject to	$2\,x_p + 3\,x_s \geq 12$
and	$4\,x_p + x_s \geq 8$
and	$x_p \geq 0$ **and** $x_s \geq 0$



$\overline{}$
10

(f) Describe in brief outline the simplex algorithm. *(4 marks)*

The simplex algorithm solves a linear programming problem. It does this in two stages. In the first stage it finds a feasible solution at a vertex. Once it has found a feasible solution it hops from vertex to vertex.

(There is a lot more to say about the simplex algorithm, but saying this much would gain full marks.)

End of question 4

$\frac{\quad}{4}$

Q4: (a) $\frac{\quad}{4}$ (b) $\frac{\quad}{4}$ (c) $\frac{\quad}{2}$ (d) $\frac{\quad}{6}$ (e) $\frac{\quad}{10}$ (f) $\frac{\quad}{4}$ Total $\frac{\quad}{30}$

END OF PAPER