DECV Algorithmics Trial Exam 1 Suggested Solutions

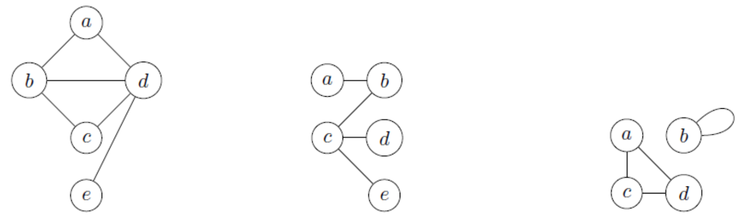
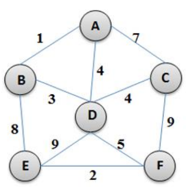
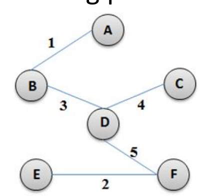
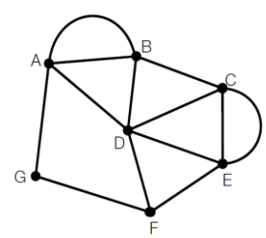
Q	Section A: Multiple Choice Questions	Topic/Answer
1	Time complexity of an algorithm is defined as: A. the relationship between the *size of the input* and the *run time* for the algorithm B. exact count of operations T(n) as a function of input size n C. The amount of time an algorithm takes to run D. The average case run time of an algorithm E. The worst case run time of an algorithm	Time Complexity A
2	Factors that affect time complexity analysis are: A. The programming language chosen to implement the algorithm B. The quality of the compiler C. The speed of the computer on which the algorithm is to be executed D. None of the above	D

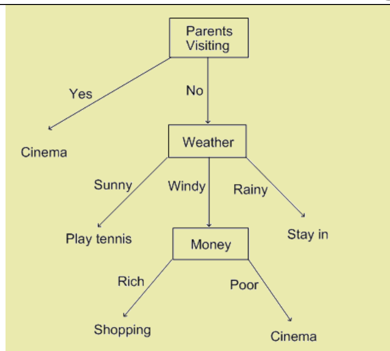
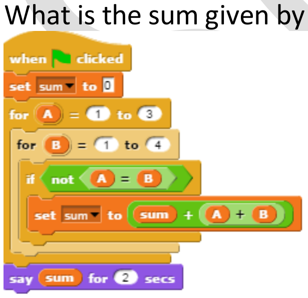
3	The time complexity for the following nested loop fragment is:
	`x := 0`
	`for j = 1 to n/2 do`
	`    for k = 1 to n*n do`
	`        x := x + j + k`
	`    end do`
	`end do`
	
	A. $O(n^2)$
	B. $O\left(\frac{n}{2}\right)$
	C. $O\left(\frac{n^2}{2}\right)$
	D. $O(n^3)$
		Outer loop executes $\frac{n}{2}$ times. For each of those times, inner loop executes n^2 times, so the body of the inner loop is executed $\left(\frac{n}{2}\right) * n^2 = \frac{n^3}{2}$ times.
		
		The algorithm is $O(n^3)$

4	The time complexity for the following nested loop fragment is:
	`x := 0`
	`for j = 1 to n do`
	`    for k = 1 to k < 3*j do`
	`        x = x + j`
	`    end do`
	`end do`
	
	A. $O(n^2)$
	B. $O\left(\frac{n}{2}\right)$
	C. $O\left(\frac{n^2}{2}\right)$
	D. $O(n^3)$
		With *dependent* nested loops: Number of iterations of the inner loop depends on a value from the outer loop
		
		When j is 1, inner loop executes 3 times; when j is 2, inner loop executes 3*2 times; … when j is n, inner loop executes 3*n times.
		In all the inner loop executes 3+6+9+…+3n = 3(1+2+3+…+n) = $3n^2/2 + 3n/2$ times.
		
		The algorithm is O(n^2).

5	Which of the following is **not** true for terms representing time complexity? A. n dominates logn B. n^2dominates n*log (n) C. n^m dominates n^k when k > m D. a^n dominates n^m for any a > 1 and m >= 0	C

6	Consider the pop operation for a Stack data structure, the time complexity of this operation is: A. O(n) B. O(1) C. O(logn) D. O(n^2)	B
7	If the number of operations does not depend on specific items, it depends only on the number of items, then the algorithm is said to be deterministic therefore: A. all possible instances of the problem ("best case", "worst case", "average case") give the same number of operations B. The worst case has the highest time complexity C. The average case is the most difficult time complexity to calculate D. The best case time complexity is said to be linear.	A
8	Let L be an empty list, the state of L after executing these operations $L \leftarrow \text{append}(L, 1)$ $L \leftarrow \text{append}(L, 5)$ $L \leftarrow \text{prepend}(L, 8)$ $L \leftarrow \text{append}(L, L)$ is: A. 1,5,8,L B. 8,L,1,5 C. 8,1,5,L D. 8,5,1,L	C
9	Consider a restaurant kitchen. When the kitchen receives orders for food they are processed in the order that they are received. Occasionally an order will need to be rushed and done ahead of the other orders. An abstract data type that could be used to model the food orders being processed by the kitchen is: A. A stack B. A list C. A queue D. A priority queue	D
10	The formal definition of the connected graphs is: A. has every pair of vertices joined by one edge. B. All vertices have a degree of 1 or higher C. There are (V-1) edges for V vertices D. All vertices have a degree of 2 or higher	B
11	The formal definition of a tree is: A. A graph that contains at least one cycle B. A graph of |V| vertices and |V-1| edges C. A connected graph with no cycles D. A graph of forests	C

12	Which of the following graphs are cyclic graphs?  (i) (ii) (iii) A. Only (i) B. Only (i) and (ii) C. Only (i) and (iii) D. Only (ii) and (iii)	C
13	A path that passes through every vertex of a graph exactly once without returning to starting vertex is an: A. Euler Path B. Shortest Path C. Minimum Cost Path D. Hamiltonian Path	D
14	Consider the following weighted Graph:  The minimum spanning tree has the cost: A. 18 B. 15 C. 14 D. 17	B. Using prims 
15	A Hamiltonian Path for the graph shown that begins at *F* is:  A. G-A-B-D-C-E-F B. F-E-C-D-B-A-G C. F-E-C-E-D-F-G-A-B-A-D-B-C-D D. F-G-A-B-D-E-F	B
16	Which data structure is used in breadth first search of a graph to hold the nodes? **A.** stack **B.** queue **C.** dictionary **D.** array	B
17	The decision tree below represents a set of activities that are done depending on whether parents are visiting :	D

	 Which action below <u>cannot</u> occur from this tree? **A.** If the parents are visiting, go to the cinema. **B.** If the parents are not visiting and it is sunny, then play tennis. **C.** If the parents are not visiting and it is windy and you're rich, then go shopping. **D.** If the parents are not visiting and it is windy and you're poor, then stay in.	
18	The signature for a Dictionary Abstract Data type looks like: $name\ Dictionary;$ $import\ key, value;$ $ops\quad newDictionary\ :\ \rightarrow dictionary;$ $insertDictionary: key \times value \times dictionary \rightarrow dictionary;$ $removeDictionary\ :\ key \times dictionary \rightarrow dictionary;$ $lookupDictionary\ :\ key \times dictionary \rightarrow value;$ Removing an item from this Dictionary has the inputs and outputs of: **A.** INPUT: key, value OUTPUT: dictionary **B.** INPUT: dictionary OUTPUT: dictionary **C.** INPUT: key, dictionary OUTPUT: value **D.** INPUT: key, dictionary OUTPUT: dictionary	D
19	The definition of Transitive closure in graph theory is: A. A directed path between two nodes. B. A True or False relation that informs if a path exists between two nodes. C. A directed acyclic graph. D. A Brute Force algorithm performed on a directed graph.	B
20	What is the sum given by the following Edgy code?  A. 54 B. 52 C. 34 D. 42	42

Section B: Extended Response Solutions

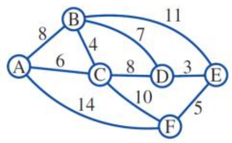
Q	Section B: Extended Response Solutions

1 Consider the following algorithm for finding the minimum spanning tree of a Graph

```
Algorithm Prims(Input Graph G, Output Tree T)
Input: G=(V,E) a weighted graph

Initialize MST:  T=(Vnew = { V0},  Enew = {} )
//Start the MST by selecting any vertex in the Graph
Repeat until Vnew = V:
      Choose minimal weighted edge {u, v} where u is in Vnew and v is not
      Add v to Vnew and {u, v} to Enew
Output: T=(Vnew , Enew )
```

Consider the following weighted graph:



a. Execute the algorithm showing the set T and E_{new} at each iteration. (2 marks)

T=(V_{new} = {A}, E_{new} = {})

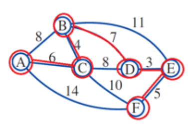
T=(V_{new} = {A,C},
E_{new} = {(A-C),})

T=(V_{new} = {A,C,B},
E_{new} = {(A-C),(C-B)})

T=(V_{new} = {A,C,B,D},
E_{new} = {(A-C),(C-B),(B-D)})

T=(V_{new} = {A,C,B,D,E},
E_{new} = {(A-C),(C-B),(B-D),(D-E)})

T=(V_{new} = {A,C,B,D,E,F},
E_{new} = {(A-C),(C-B),(B-D),(D-E),(E-F)})

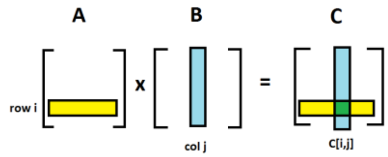


b. Are there any non trivial loop invariants that you can identify? List and describe. (2 marks)
the repeat until loop has a loop invariant that is maintained, which is that the tree being formed T is an MST for the subset of new vertices at each loop which is always the MST

c. Show the correctness of the algorithm in finding the Minimum Spanning Tree. (2 marks)
- The Tree T is grown by selection of an edge that is the minimum weighted edge emanating from the current T until all vertices are visited.
- By using this strategy on each iteration minimum weighted edges will be selected to connect all vertices, which will give the MST

d. What is the algorithm design pattern used by Prim's Algorithm? Describe the general properties of this design pattern.
- Greedy Algorithm
- At each junction the options are examined and the most immediately favourable one is selected

2 Given two n x n matrices A and B, find the time complexity of the algorithm below for computing their product



C=AB.

```
Algorithm MatrixMult (Input matrices: A (nxn), B (nxn),  Output matrix: C(nxn))
// multiply two square matrices and gives result C
// the conventional notation for describing
// each element of matrix is described by its row,column
// For example A[1,3] is the element in row 1, column 3
for i=1 to n do
        for j=1 to n do
                for k=1 to n do
                        C[i,j]:=C[i,j]+A[i,k]*B[k,j]
                end do
        end do
end do
return C
end Algorithm
```

a. Complete the required actions using conventional notation for this algorithm to work correctly.
   C[i,j]:=C[i,j]+A[i,k]*B[k,j]

b. What is the time complexity of the algorithm MatrixMult? Show all your reasoning for your answer.

There are three nested loops nxnxn operations O(n^3) in more detail the number of multiplications can be expressed by the triple sum. Where k is a constant

c. Compare the time complexity for the best, average and worst cases for this algorithm.

Same for all cases as fixed operations are not dependent on values of input only the size of the input

$$T(n) = k\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} 1$$

If we start with the innermost sum

$$\sum_{k=1}^{n} 1 = n$$

Our sum is reduced to a double sum.

$$T(n) = k\sum_{i=1}^{n}\sum_{j=1}^{n} n$$

this innermost sum can be reduced to:

$$\sum_{j=1}^{n} n = n^2$$

Our time complexity

$$T(n) = k\sum_{i=1}^{n} n^2$$

$$=> T(n) = kn^3 = O(n^3)$$

Since this is the same amount of operations done for all types of input and has no conditional variation for input data, then the time complexity is determined only on the size of the input matrix, then it is the same for best case, worst case and average case time complexities.

3 A Transport plane has to deliver the most valuable set of items to a remote location without exceeding the plane's capacity.

There are n items that can be selected

item	1	2	3	....	i		n
weight	w1	w2	w3		wi		wn
value	v1	v2	v3		vi		vn

a. In how many ways can the n items be selected? Explain your reasoning.
   - An item can be included or not included so there are 2 states for each item
   - Since there are n items there are 2^n possible ways of filling the plane

b. What is the classification of this type of problem?
   - This problem is an NP-Hard or NP-Complete problem
   - This is because of the combinatorial explosion of choices and the exponential time complexity of deciding how to fill the plane
   - This problem is essentially the same as the knapsack problem.

c. Describe the different classes of problems that are defined in Computer Science. What criteria determine in which class a particular problem belongs to.
   - P (Polynomial) class problems have algorithms with polynomial time complexity
   - NP (Non-deterministic Polynomial) class problems have exponential time complexity for their quickest algorithms, but their solutions if found can be checked in polynomial time for correctness
   - NP-Complete problems are NP problems that should a solution be found for one then essentially they are all solved in the same manner
   - NP-Hard problems include NP-Complete problems and other problems that do not have any known algorithms as such.

4 **Question 4 (8 marks)**

Consider a ternary search. This is an algorithm for searching for a key value K in a sorted array A[1..n]. If n=1, compare element with search key K, otherwise search recursively by comparing K with range of A[1..n/3] if larger compare K with A[n/3..2n/3] if larger still compare K with A[2n/3…n]. Search for K in the subset of A that has the appropriate range.

a. What design technique is this algorithm based on? Describe the principles of this design pattern. (2 marks)
   - Divide and conquer, the data is split into 3 parts and value , K is located in one of the parts
   - The principles of divide and conquer are that the problem can be split up into independent subsets, the solution is found in the subsets if possible and merged into an overall solution

b. Write out the algorithm in pseudocode. (3 marks)

```
function ternary_search(input A[], input key, input imin, input imax)
 // test if array A is empty imin is the lower bound, imax is the upper bound
        if (imax < imin) then
         // set is empty, so return value showing not found
         return KEY_NOT_FOUND;
        else
            // cut set into thirds
          ithird = round((imax+1-imin)/3)
        end if

    if (A[imin+ithird-1] > key)
     // key is in lower third
```

```
            return ternary_search(A, key, imin, imin+ithird);
        else if (A[imin+2*ithird-1] > key)
            // key is in middle third
            return ternary_search_search(A, key, imin+ithird, imin+2*ithird-1);
        else if (A[imax] < key)
            // key is in upper third
            return ternary_search_search(A, key, imin+2*ithird, imax);
        else
            // key has been found at position
            return imin+ithird
        end if
    end function
```

Set up a recurrence relation for the time complexity of the algorithm. (1 marks)

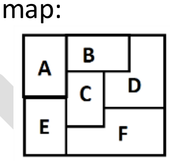
$$T(n) = T\left(\frac{n}{3}\right) + O(1)$$

c. Find the worst case time complexity for this algorithm. How does this algorithm compare with the time complexity of Binary Search? (2 marks)

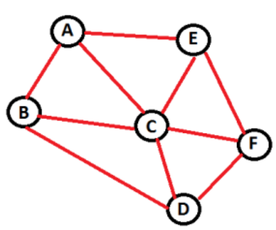
- O(logn)
- Same as binary search, base of log is irrelevant for time complexity analysis using big O notation

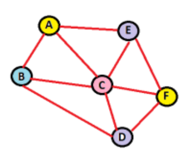
5 **Question 5 (10 marks)**

Consider the following map:



a. Represent the information shown in the diagram above as a connected graph with nodes and edges. (2 marks)





b. (i) Explain how we can use the graph-colouring problem to colour the map so that no two neighbouring regions are coloured the same. (2 marks)

Once the regions are represented as a graph, the nodes must be coloured with the minimum count of colours so that no two adjacent nodes have the same colour.

(ii) What is the minimum number of colours required in this instance? (1 marks)

4 colours is the minimum

(iii) Write a greedy algorithm in pseudocode to colour the map so that no two neighbouring regions are the same colour. (3 marks)

```
Algorithm: Greedy Graph Colour
input  : A graph G
output: An approximate number of colours required to colour G
begin
    Let mincolours be an approximation for the minimum number of colours
    required to colour G
    Initialise mincolours to 1
    while any nodes of G are uncoloured do
        Select an uncoloured node from G
        if the selected node could be coloured with an available colour then
            Colour the selected node with the lowest ordered available colour
        else
            Increment mincolours by 1
            Colour the selected node with the new colour
```

(iv) What is the time complexity of your algorithm? (2 marks)

Each vertex needs to compare its colour to every uncoloured neighbouring vertex, the while loop drives the algorithm O(|V|)

6 **Question 6** – (14 Marks)

a. Explain how one can identify connected components of a graph by using:

i. Depth-first search (2 marks)

DFS can be used to traverse a graph. One starts at the root of a tree or by selecting some arbitrary node as the root in the case of a graph and explores as far as possible along each branch or path before backtracking. This can be iterative or recursive process until target is found or all options are exhausted and target is not in the graph or tree.

ii. Breadth-first search (2 marks)

BFS can be used to traverse a graph. It starts at the tree root or some arbitrary node of a graph and explores all the neighbor nodes first, before moving to the next level neighbors. This can be iterative or recursive and therefore can find a path to a target if one exists.

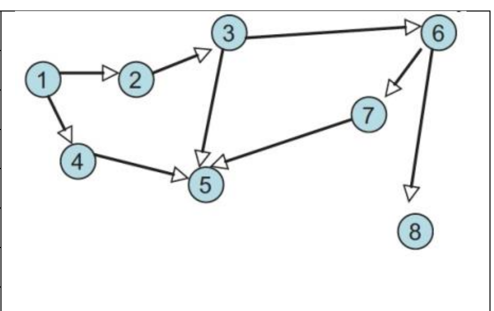
b. Explain how one can check a graph's acyclicity by using Breadth-first search. (2 marks)

For BFS neighbouring vertices are clearly marked at most once, added to the queue at most once (since that happens only when it's marked), and therefore removed from the queue at most once. If there is an adjacency from a node to a previously visited node then it is possible that a cycle or an acyclicity exists. If the previously visited node is a sink (ie has no outgoing edges) then the graph is acyclical.

c. Consider the directed graph shown above:

(i) Show the order of nodes visited by Depth-First Search starting at node 1 and ending at node 8. (Always select in numeric order when given multiple options.) (2 marks)

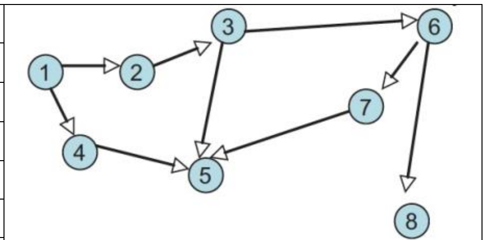
Nodes Visited	DFS Stack
1	2, 4
1, 2	3, 4
1, 2, 3	5, 6, 4
1, 2, 3, 5	6, 4
1, 2, 3, 5, 6	7, 8, 4
1, 2, 3, 5, 6, 7	8, 4
1, 2, 3, 5, 6, 7, 8	Stop target found



(ii) Show the order of nodes visited by Breadth-First Search starting at node 1 and ending at node 8.

(Always select in numeric order when given multiple options.) (2 marks)

Nodes Visited	BFS Queue
1	2, 4
1, 2	4, 3
1, 2, 4	3, 5
1, 2, 4, 3	5, 6 (5 already marked)
1, 2, 4, 3, 5	6
1, 2, 4, 3, 5, 6	7, 8
1, 2, 4, 3, 5, 6, 7	8
1, 2, 4, 3, 5, 6, 7, 8	Found target



d. Write an algorithm in pseudocode that will perform a topological sort on a directed graph. (3 marks)

Algorithm A	OR Algorithm B
Removing the Source en route to the sink	based on recursive DFS

Algorithm A

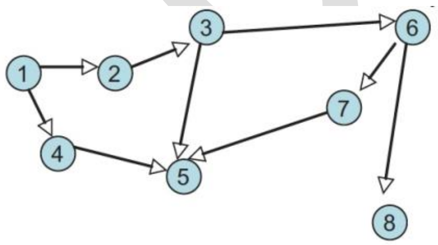
```
L ← Empty list that will contain the sorted elements
S ← Set of all nodes with no incoming edges
while S is non-empty do
   remove a node n from S
   add n to tail of L
   for each node m with an edge e from n to m do
      remove edge e from the graph
      if m has no other incoming edges then
         insert m into S
if graph has edges then
   return error (graph has at least one cycle)
else
   return L (a topologically sorted order)
```

Algorithm B

```
L ← Empty list that will contain the sorted nodes
while there are unmarked nodes do
   select an unmarked node n
   visit(n)

function visit(node n)
   if n has a temporary mark then stop (not a DAG)
   if n is not marked (i.e. has not been visited yet) then
      mark n temporarily
      for each node m with an edge from n to m do
         visit(m)
      mark n permanently
      unmark n temporarily
      add n to head of L
```

e. Demonstrate your algorithm on the directed graph shown above. (1 mark)



S={1} L={}

S={2,4} L={1}

S={4, 3} L={1,2}

S={3} L={1,2,4}

S={6} L={1,2,4,3}

S={7,8} L={1,2,4,3,6}

S={8,5} L={1,2,4,3,6, 7}

S={5} L={1,2,4,3,6,7,8}

L={1,2,4,3,6,7,8,5} is one possible order

7 **Question 7** – (16 Marks)

a. Given the following list of integers 66, 33, 40, 22, 55, 88, 60, 11. Show the stages of ordering produced by the Bubblesort algorithm to sort these integers. (2 marks)

66, 33, 40, 22, 55, 88, 60, 11	66 > 33 bubble 66 along
33, 66, 40, 22, 55, 88, 60, 11	
33, 40, 66, 22, 55, 88, 60, 11	
33, 40, 22, 66, 55, 88, 60, 11	
33, 40, 22, 55, 66, 88, 60, 11	66 < 88, so bubble 88
33, 40, 22, 55, 66, 60, 88, 11	
33, 40, 22, 55, 66, 60, 11, 88	88 in correct place
33, 22, 40, 55, 66, 60, 11, 88	
33, 22, 40, 55, 60, 66, 11, 88	
33, 22, 40, 55, 60, 11, 66, 88	66 in correct place
22, 33, 40, 55, 60, 11, 66, 88	
22, 33, 40, 55, 11, 60, 66, 88	60 in correct place
22, 33, 40, 11, 55, 60, 66, 88	55 in correct place
22, 33, 11, 40, 55, 60, 66, 88	40 in correct place
22, 11, 33, 40, 55, 60, 66, 88	33 in correct place
11, 22, 33, 40, 55, 60, 66, 88	22 in correct place

apologies this list had a few too many elements ☹

b. Given the following list of integers 66, 33, 40, 22, 55, 88, 60, 11. Trace by hand the Quicksort algorithm that uses the **leftmost element** as the pivot to sort these integers. (2 marks)

66, 33, 40, 22, 55, 88, 60, 11	
33, 40, 22, 55, 60, 11, 66, 88	pivot placed in correct position
33, 40, 22, 55, 60, 11, 66, 88	recursive sort before pivot, sort after pivot
22, 11, 33, 40, 55, 60, 66, 88	pivot placed in correct position
11, 22, 33, 40, 55, 60, 66, 88	pivot placed in correct position

c. Consider sorting the following of list n=8 items 10, 9, 8, 7, 6, 5, 4, 3, using Quicksort with the pivot at the **leftmost element.**

(i) How many actions would be needed to sort this list? (2 marks)

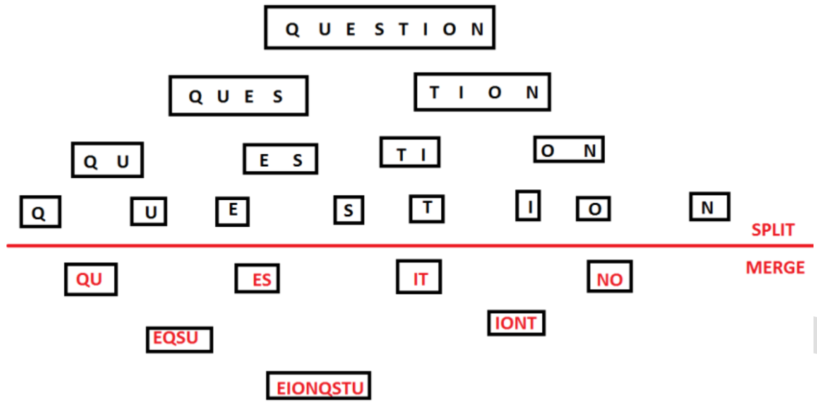
10, 9, 8, 7, 6, 5, 4, 3
9, 8, 7, 6, 5, 4, 3, 10
8, 7, 6, 5, 4, 3, 9, 10
7, 6, 5, 4, 3, 8, 9, 10
6, 5, 4, 3, 7, 8, 9, 10
5, 4, 3, 6, 7, 8, 9, 10
4, 3, 5, 6, 7, 8, 9, 10
3, 4, 5, 6, 7, 8, 9, 10

Actions needed are 9+8+7+6+5+4+3+2+1
Pivot selected is not dividing the data into halves, count of comparison operations given by:
$\sum_{i=1}^{n} n-i$ => O(n^2

(ii) What strategy could be used to improve the performance of Quicksort for all cases of input? (2 marks)

- A random pivot could be selected at each iteration
- Selection of a random value will effectively halve most input even if it is ordered in reverse or some other extreme way.

d. Show the splits and the merges done when the recursive Mergesort is used to sort the letters in the list {Q, U, E, S, T, I, O, N}. (2 marks)



e. Consider the following algorithm for Insertion Sort, that accepts as input an array A of values that need to be sorted. The array A has n elements and each element is referenced by the notation A[1]....A[n].

```
for i := 2 to length(A)
   j := i
   while j > 1 and A[j-1] > A[j]
      swap A[j] and A[j-1]
      j := j - 1
   end while
end for
```

(i) What is the run time complexity of this algorithm, assuming that the "swap" has a constant count of commands and has the order O(1)? Show all your reasoning. (2 marks)

- outer loop runs for n-1 times
- inner loop commands are executed 1 + 2 + 3 + 4 +....n-1 times $\sum_{i=1}^{n-1} i = \frac{(n-1)(n-2)}{2}$
- $O(n^2)$

(ii) Identify all loop invariants in this algorithm? (2 marks)

- trivial: i<= length(A)
- non-trivial: A[j-1] < A[j]

(iii) Give a justification for the correctness of this algorithm in sorting values. (2 marks)

as the inner loop runs the non-trivial loop invariant: A[j-1] < A[j] as this holds for j=2 to the length(A) and this results in the array A being sorted by this algorithm, and shows it is correct on completion

8 **Question 8** (12 marks)

Describe the following testing methodologies and their elements and how they are used to test algorithms.

a. Black Box (2 marks)

The system is tested without knowledge of it internal structure, hence it is treated as a black box. Tests include pairwise testing, boundary testing, edge testing and error guessing.

b. White Box (2 marks)

Knowledge of the programming environment and internals of the system are required for white box testing. The internal code of the system is tested to check that all paths are covered in the code as well as all logical combination test of conditional statements.

c. The following inputs determine the information required to book a flight.

- Airline Type = {**S**cheduled, **L**ow Cost, **C**harter}
- Cabin= {**F**irst Class, **B**usiness, **E**conomy}
- Fare Type={**O**ne Way, **R**eturn}
- Fare Conditions = {**F**lexible, **R**estricted}

(i) What are the total possible combinations of booking a flight using these inputs? (1 mark)

3x3x2x2=36 ways

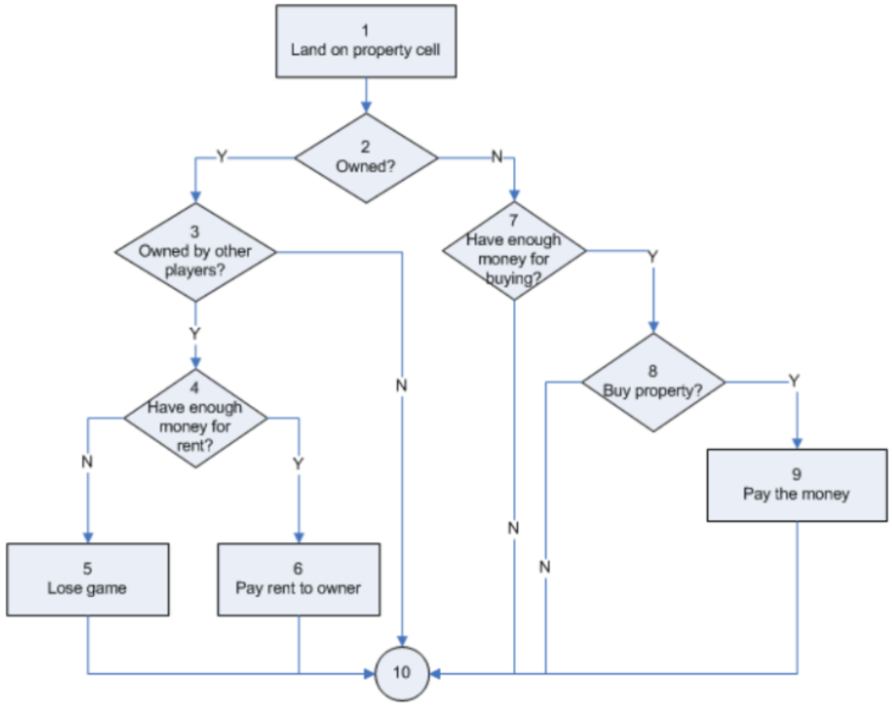
(ii) Demonstrate how pairwise testing can be used for this example, showing how many tests will result using this method. (2 marks)

Pairwise Tests	Airline	Cabin	Fare Type	Restrictions
1	Sched	First	One Way	Flexible
2	Sched	Business	Return	Restricted
3	Sched	Economy	One Way	Flexible
4	Low Cost	First	Return	Restricted
5	Low Cost	Business	One Way	Restricted
6	Low Cost	Economy	Return	Flexible
7	Charter	First	One Way	Restricted
8	Charter	Business	Return	Flexible
9	Charter	Economy	One Way	Restricted

d. What is boundary and edge testing? Give an example of each type of this testing. (2 marks)

- Boundary testing is making sure that input values are within a defined domain; for example, a person's age has to be between 0 and 120 years old, therefore test 80 years should return ok, also test outside domain eg. 140 years should return false.
- Edge testing is making sure that inputs nearest to intersecting boundaries are correct; for example, a person's age is tested at -1,0,1 and 119,120,121 to test the edges for age input.

Consider the following flowchart for the steps involved in purchasing a property.



Enumerate all the possible paths through this flowchart, using the numbers on the nodes. How many possible paths will need to be tested? (3 marks)

There are 6 possible paths through this code to get from node 1 to node 10.

1. 1-2-3-4-5-10 (property owned by others, no money for rent)
2. 1-2-3-4-6-10 (property owned by others, pay rent)
3. 1-2-3-10 (property owned by the player)
4. 1-2-7-10 (property available, don't have enough money)
5. 1-2-7-8-10 (property available, have money, don't want to buy it)
6. 1-2-7-8-9-10 (property available, have money, and buy it)