Programming 2 Exam Paper

Q1 E i

```
LLNODE appendIfNotPresent (LLNode head, int data){

 if (isEmpty){                                1
  this.head = head;                           1
  head.data = data;                           1
  return                                      1
 }
 else {
   while (current.next != null){              N
     if (current.data = data){                N
       return                                 1
     }
   }
   current.next = head;                       1
   head.data = data;                          1
 }
     = 2N + 6.
```

2 a It is possible to implement a Stack structure using eithe a linked list or a resizing array implement. Thus we want to know which implementation is more efficient

Asymptotic cost is calculated by firstly dropping coefficien of N's. Take the highest order of N this is cost.

Amortized cost is Asymptotic costs / N, it is the cost of N operations over time, not just one operation, thus there is a difference

Asymptotic gives us a long run cost or the algorithm but is not precise.

4

Amortized gives the cost of running algorithm mainly, which accounts for all difference in operations

b Resizing array implementation of a Stack push/pop operation takes O(1) amortized time, push/pop operation takes O(n) in olympic terms.

Over period of use the resizing will not occur at each move, thus it will tend towards O(1)

C Must satisfy:
- No node has two red links connect to it.
- Every path from root to null link has same number of black links
- Red links always lean left
- links go red-black-red etc

D This is the actual height of the tree if it were a simple binary search tree;

6
```
Public Key Ceiling (Key key){
    Node x = ceiling(root, key);
    if (x = null) return null;
    else return x.key
}
Private Node ceiling( Node x, Key key){
    if (x == null) return null;
    int cmp = key. CompareTo(x.key);
    if (cmp = = 0) return x;
    if (cmp > 0) return ceiling (x.right, key);
    Node t = ceiling(x. left, key);
    if (t != null) return t;
    else return x;
}
```

5.

[illegible]amming 2 Exam Paper [illegible]

[illegible]-property: A cut in a graph is a partition of the vertices into two (nonempty) sets

[illegible] property - Given any cut, the crossing edge of min weight is in the MST

[illegible] cutting starting at (6)

4 → 7 0.2
7 → 5 0.3
7 → 6 0.4
4 → 3 0.2
3 → 2 0.1
4 → 1 0.2

[illegible] Sort edges i to j in ascending order of weight
Add the next edge to the tree T unless doing so would create a cycle
Continue until all nodes are connected

Solution to the [illegible]

2 → 3 0.1 ✓
1 → 4 0.2 ✓
3 → 4 0.2 ✓
3 → 4 0.2
4 → 7 0.2
2 → 1 0.3
2 → 4 [illegible]

cycle
✓
✗

D i. -Start at start point
-pick next neighbour arbitrarily and mark as visited
put other neighbours on stack
- Get neighbours of neighbour
- Select one add to visit to stack, continue

ii. 0 → 1 → 3 → 5 → 4 → 7 → 9 6, 8 ·2

E i. This is the shortest path from a vertex S to every other vertex.

Initialise distTo[s] = 0 and distTo[v] = ∞ for all other vertices
Repeat until optimality condition are satisfied
-non negative weights

5.

Programming 2 Exam Paper Soln

3 a) Cut-property: A cut in a graph is a partition of all vertices into two (nonempty) sets.

Cut propert - Given any cut, the crossing edge must be of a min weight and in the MST.

b By cutting Starting at 4

4 → 7 0.2
7 → 5 0.3
7 → 6 0.4
4 → 3 0.2
3 → 2 0.1
4 → 1 0.2

c) Rank the edges i to j in ascending order of weight.
Add the next edge to the tree T unless doing so would create a cycle
Continue until all nodes are connected

ii. 2 → 3 0.1 ✓ Solution is the same
1 → 4 0.2 ✓
3 → 4 0.2 ✓
4 → 7 0.2 ✓
2 → 1 0.3 X cycle
2 → 4 0.3 X cycle
5 → 7 0.3 ✓
4 → 5 0.4 X cycle
5 → 6 0.4 ✓
6 → 7 0.4 X