COMP108 Algorithmic Foundations

Tutorial 9 (Suggested solution and Feedback)

w/c 24th April 2017

1. (a) preorder: f, b, a, d, c, e, g, k, h

   (b) inorder: a, b, c, d, e, f, g, h, k

   (c) postorder: a, c, e, d, b, h, k, g, f

2. The first player removes 1 coin. Then whenever the second player removes x coins, the first player removes $5 - x$ coins. Then the first player will remove the last coin and win.

   If there are 25 coins initially, there is no winning strategy for the first player. Instead, the second player has a winning strategy.

   In general, if the number of coins is NOT a multiple of 5, then the first player has a winning strategy; otherwise, the second player has a winning strategy.

3.

```
public void bfs(int vindex) {
    int[] list = new int[n];
    // head & tail point to the head & tail of the list
    int head=0, tail=0, next, j;

    // mark the first vertex and print its label
    vertex[vindex].mark = true;
    System.out.print(vertex[vindex].label + " ");

    // append the first vertex to tail of the list
    list[tail] = vindex;
    tail++;

    // while the list is not empty (tail > head), traverse
    while (tail > head) {
        // get the next vertex from the head of the list
        next = list[head];
        head++;
        // for every unmarked neighbor of this vertex,
        // mark it, print its label and insert it to tail of list
        for (j=0; j<n; j++) {
            if ((adj[next][j] == 1) && !vertex[j].mark) {
                vertex[j].mark = true;
                System.out.print(vertex[j].label + " ");
                list[tail] = j;
                tail++;
            }
        }
    }
}
```

```
// recursive dfs from the vertex with index vindex
public void dfs(int vindex) {
    int j;

    // print the label of the current vertex and mark it
    System.out.print(vertex[vindex].label + " ");
    vertex[vindex].mark = true;

    // for every unmarked neighbor, recursively call dfs()
    for (j=0; j<n; j++) {
        if ((adj[vindex][j]==1) && !vertex[j].mark) {
            dfs(j);
        }
    }
}
```