a)

T	O	B	Y	R	E	A	D	M	A	N

b)

20	15	2	25	18	5	1	4	13	1	14

c)
XYZ = 118

d)
YZ + 1 = 19

e)
line 19, word 5 = occasionally

f)
provide, technical, assistance, read, comment, draft, report, give, guidance, standard, amount, work, required, this, last.

g)
prov, tech, assi, read, comm, draf, repo, give, guid, stan, amou, work, requ, this, last.

1.

Toby Readman – treadm01@dcs.bbk.ac.uk

2.
amou, last, guid, give, draf, comm, assi, requ, stan, repo, read, this, work, tech, prov.

3.
prov, assi, amou, comm, draf, give, guid, last, tech, read, repo, stan, requ, work, this.

Assuming that nodes are numbered by the preorder traversal above, that is 3rd node is *amou* and 6th node is *give*, etc.

Stack after 3rd:

assi
prov

After 6th:

prov

After the 9th node is visited the stack is empty:

⌴

After 12th:

tech

4.
Delete *prov*

Delete *tech*

Delete *assi*

5.

6.

```
QUEUE (F, R, y)
        if (F ≠ nil and LAST ≠ M)  {// if list is not empty and last node in list is not full
                LAST++; //Increase last index by one
                R↑ITEM[LAST] ← y; // add the element y to position new last position
        }
        else {
                P ← GETNODE();
                if (F = nil) { // if list is empty
                        FIRST ← 1; // assign initial FIRST and LAST indexes
                        LAST ← 1;
                        P↑ITEM[FIRST] ← y; // assign the element to first position in node
                        F↑LINK ← P;
                        R↑LINK ← F.ITEM; // rear link links to first node
                }
                else if (LAST = M) { // list is not empty but last node is full
                        LAST ← 1; // assign new LAST index
                        P↑ITEM[LAST] ← y;
                        R↑ITEM↑LINK ← P; // assign link of current last node to new node
                        R↑LINK ← P; // assign rear link to new node
                }
        }
```

```
UNQUEUE(F, R)
        if (F = nil) {
                deal_with_UNDERFLOW();
        }
        else if (FIRST = M) { // If the node has only one element, i.e. the node will be removed
                P ← F↑ITEM;
                if (F↑LINK = R↑LINK) { // if there is only one node in the list assign nil to F
                        F↑LINK ← nil;
                }
                else {
                        F↑LINK ← P↑LINK; // assign current second node to F
                }
                result ← P[FIRST]; // get element from node being unqueued
                release(P);
                FIRST ←1; // assign new FIRST index
                return result;
        }
        else { // if the element at FIRST is not the final element of F↑ITEM
                result = P[FIRST];
                P[FIRST] ←nil; // delete element from array
                FIRST++;
                return result;
        }
```