Q1.

```
def visitall (G):
    visit(v1,1)

def visit(node v, clock t):
  v.pre=clock
  clock++

  mark v
  v.low = v.pre

  for each edge v-w:
    if v is marked:
        v.low = min(v.low, w.pre)
    else:
        t = visit(w, clock)
        clock = t
        v.low = min(v.low, w.low)

    return clock
```

No point if global variable(s)/stack(s) were used.
Points deducted for not updating clock correctly and not ensuring that v1.pre=1.

Q2.
(continued after the proof given in question)
Let v be one vertex to which there is no path from u. Since G is semi-connected, there must be a path from v to u.

There can be two cases:
(a) v is visited before u during DFS. Since there is path from v to u, so u will be discovered during DFS(v) and hence, u will be finished before v is finished. So, finish(u) < finish(v) leading to a contradiction of the fact that finish(u) is largest.
(b) u is visited before v during DFS. Since there is no path from u to v, when u is finished v will still be unexplored. Hence, v will be visited after u is finished, and therefore, finish(v) > finish(u).
This also contradicts the fact that finish(u) is largest.

Point deducted for using "u is the root or one of the roots of the DFS traversal" without explaining any reason.

Q3. Easy counter-example.

Q4. (a) Yes (b) No (c) k' = |V| - 2k  (d) Yes (since you can easily show that HalfClique is NP-complete based on the facts in Q4.)

Q5.

```
def Reduce3SAT_Clique(F):
  k = m
  Construct G in the following manner:
    for every clause (la OR lb OR lc), where la, lb, lc are literals:
      add there vertices, one for each literal
    for every pair of vertices (vi, vj):
      if vi and vj belong to the same `clause': do not add any edge
      if vi and vj correspond to a variable and its negation: do not
add any edge
      else, add an edge between vi and vj
    return (G,k)
```

(i) Number of vertices in G = 3m
(ii) Number of edges in G <= 9m(m-1)/2 = (3m choose 2) - 3m
(iii) k=m
(iv) Output of reduction : simple