**PG 175**

**Code:**

**algorithm** *GenericSearch*(*G*,*s*)

⟨ ***pre-cond*** ⟩***:*** *G* is a (directed or undirected) graph, and *s* is one of its nodes.  
⟨ ***post-cond*** ⟩***:*** The output consists of all the nodes *u* that are reachable by a path in *G* from *s*.

begin

*foundHandled* = ∅

*foundNotHandled* = {*s*}

loop

⟨***loop-invariant***⟩***:*** See LI1, LI2.  
exit when *foundNotHandled* = ∅  
let *u* be some node from *foundNotHandled*

for each *v* connected to *u*

if *v* has not previously been found then

add *v* to *foundNotHandled*

end if

end for

move *u* from *foundNotHandled* to *foundHandled*

end loop

return *foundHandled*

end algorithm

**PG 190**

**Code:**

**algorithm** *DepthFirstSearch*(*G*,*s*)

⟨ ***pre-cond*** ⟩***:*** *G* is a (directed or undirected), graph, and *s* is one of its nodes.

⟨ ***post-cond*** ⟩***:*** The output is a depth-first search tree of *G* rooted at *s*.

begin

*foundHandled* = ∅  
*foundNotHandled* = {⟨*s*, 0⟩}  
*time* = 0 % Used for *time stamping*. See following discussion.

loop

⟨***loop-invariant***⟩***:*** Seeprecedinglist.

exit when *foundNotHandled* = ∅

pop ⟨*u*, *i*⟩ off the stack *foundNotHandled*

if *u* has an (*i*+1)st edge ⟨*u*, *v*⟩

push ⟨*u*, *i* + 1⟩ onto *foundNotHandled*

if *v* has not previously been found then

π(*v*) = *u*⟨*u*, *v*⟩ is a tree edge  
push ⟨*v*, 0⟩ onto *foundNotHandled*

*s*(*v*) = *time*; *time* = *time* + 1

else if *v* has been found but not completely handled then

⟨*u*, *v*⟩ is a back edge

else

(*v* has been completely handled)

⟨*u*, *v*⟩ is a forward or cross edge

end if

else

move *u* to *foundHandled*

*f* (*v*) = *time*; *time* = *time* + 1

end if

end loop

return *foundHandled*

end algorithm