1.

What's the meaning of:

pid, ppid? process id, parent process id

status ? PROC status is FREE or READY … ETC

priority ? priority of when the process is scheduled

event ? value to sleep on

exitCode ? value returned when the process exits

READ 3.5.2 on Process Family Tree. What are the PROC pointers child, sibling, parent used for?

These are used to keep track of where each process is created from and where forks occur

5.

fork : READ kfork() on Page 109: What does it do?

creates a new child task then put it in the ready queue

switch : READ tswitch() on Page 108: What does it do?

switch takes in a process as a input then returns a new process almost like a switch box

exit : READ kexit() on Page 112: What does it do?

free's the task inputted and all of its children then awakens the parent of the task inputted

sleep : READ ksleep() on Page 111: What does it do?

for the process inputted ksleep puts it to sleep then free's the cpu

wakeup : READ kwakeup() on Page 112: What does it do?

finds all processes sleeping on an event value and wakes them up

wait : READ kwait() on Page 114: What does it do?

finds the pid of the process being waited on then returns it

6.

While P1 running, enter fork: What happens?

Process 2 moves from the freelist to readQueue

Enter fork many times; How many times can P1 fork? 7 times **WHY?** there are 7 processes in the freeList to start.

7.

Switch to run P2. Where did P1 go?

p1 is now the parent of p2

WHY?

p2 is a forked off of p1

P2: Enter sleep, with a value, e.g.123 to let P2 SLEEP.
What happens?
p2 sleeps waiting on event code #123
WHY?
it was put to sleep with that code
Now, P1 should be running. Enter wakeup with a value, e.g. 234
Did any proc wake up?
no
WHY?
no processes are sleeping on code #234

P1: Enter wakeup with 123
What happens?
p2 awakens
WHY?
p2 slept on code 123 and when we enter wake up 123 it wakes up all processes sleeping on code 123

8.

P1: enter wait; What happens?
nothing
WHY?
p1 has no children

CASE 1: child exit first, parent wait later
P1: fork a child P2, switch to P2.
P2: enter exit, with a value, e.g. 123 ==> P2 will die with exitCode=123.
Which process runs now?
p1
WHY?
p1 is the parent of the process that we killed
enter ps to see the proc status: P2 status =
zombie

(P1 still running) enter wait; What happens?
p1 waits until its child is set to zombie status then cleans up
enter ps; What happened to P2?
p2 was freed

CASE 2: parent wait first, child exit later
P1: enter fork to fork a child P3
P1: enter wait; What happens to P1?
p1 sleeps

WHY?
because it's waiting for p3 to become a zombie
P3: Enter exit with a value; What happens?
p3 is now a zombie and is instantly cleaned up from p1 who was waiting for it to become zombie
P1: enter ps; What's the status of P3?
free
WHY?
p1 cleaned it up

9.
P1 should be running WHY?
when p2 exits we switch to the parent process which is p1
P1: enter ps to see proc status: which proc is ZOMBIE?
p2
What happened to P2's children?
nothing they are waiting ready to complete a task
P1: enter wait; What happens?
p1 cleans up the zombie's
P1: enter wait again; What happens?
process changes to p3
WHY?
p1 is waiting for p3 to exit so p3 is running now

How to let P1 READY to run again?
exit the children of p1 then let them get cleaned up by waiting from p1