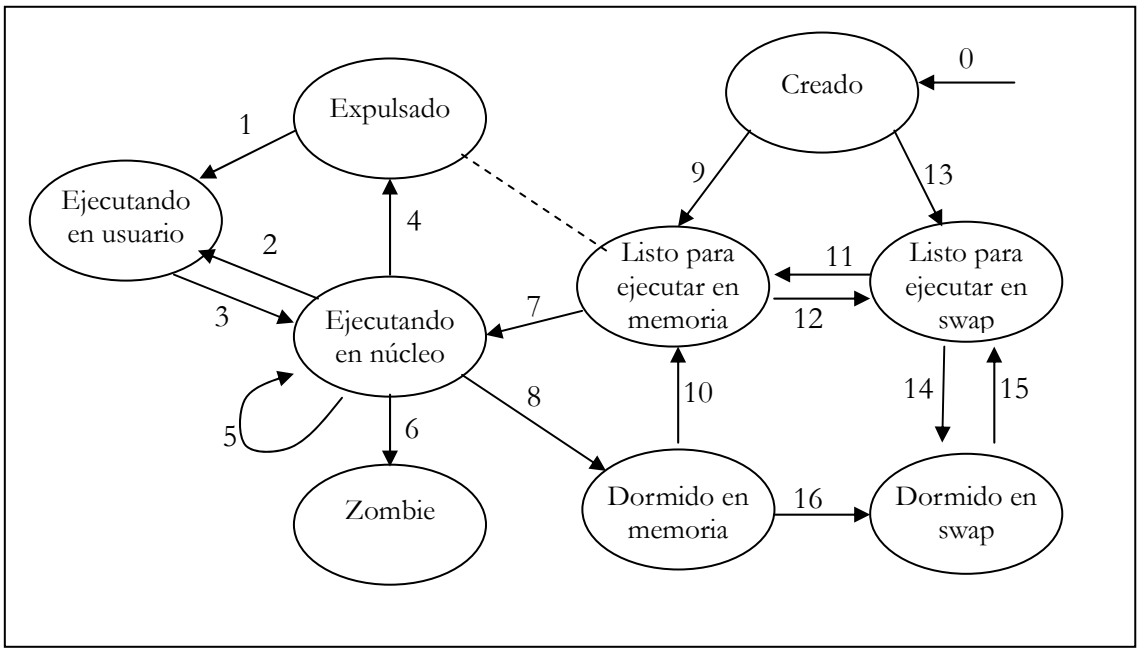
EJERCICIO 2 (4 Puntos) — Tiempo estimado: 60 Minutos

Tenemos un sistema operativo de tipo UNIX en el que los procesos pueden estar en uno de los nueve estados que se muestran en la figura, donde además aparecen las posibles transiciones entre estados. El núcleo utiliza una política de planificación por prioridades no expulsora.
El gestor de memoria utiliza intercambio con particiones variables. Este gestor lleva la cuenta del espacio libre y ocupado, asignando el espacio libre mediante el algoritmo el primero que sirva. Para gestionar el área de swap existe un proceso swapper que se encarga de intercambiar procesos entre memoria principal y memoria secundaria (área de swap).



Estado	Significado
Ejecutando en usuario	Ejecutando en modo usuario
Ejecutando en núcleo	Ejecutando en modo núcleo
Listo para ejecutar en memoria	Listo para ejecutar tan pronto como el núcleo lo planifique
Dormido en memoria	No puede ejecutar hasta que ocurra un evento; proceso bloqueado en memoria principal
Listo para ejecutar en swap	El proceso está listo para ejecutar, pero el proceso swapper debe cargar el proceso en memoria principal antes de que el núcleo pueda planificarlo para su ejecución
Dormido en swap	El proceso está esperando un evento y ha sido expulsado al almacenamiento secundario (estado de bloqueado)
Expulsado	El proceso ha regresado de modo núcleo a modo usuario, pero el núcleo lo ha expulsado y ha realizado la activación de otro proceso
Creado	El proceso ha sido creado recientemente y aún no se le ha asignado memoria. No está listo para ejecutar
Zombie	El proceso ya no existe, pero deja un registro para que lo recoja su proceso padre

Los estados “Expulsado” y “Listo para ejecutar en memoria” se gestionan con una misma cola de procesos. Simplemente sirven para diferenciar el motivo por el cual el proceso ha llegado a la cola de procesos que están preparados para ejecutar.

Se pide:

1.- De las transiciones 8, 14 y 16, indique cuál es incorrecta, en el sentido de que es imposible que se produzca. Razone la respuesta.

2.- Supongamos que el proceso swapper sigue los siguientes criterios para cargar procesos en memoria principal:

1. Cada vez que se activa solo intenta cargar el proceso más prioritario de los que están en estado "Listo para ejecutar en swap".
2. En primer lugar intenta cargarlo en algún hueco libre que haya en la memoria (no se realiza compactación).
3. Si no ha sido posible el paso 2, intenta expulsar a uno o varios procesos que estén en estado "Dormido en memoria" para dejar hueco suficiente en la memoria.
4. Si no ha sido posible el paso 3, expulsa a uno o varios procesos que estén en estado "Listos para ejecutar en memoria" para dejar hueco suficiente en la memoria. En este caso nunca expulsará a un proceso que sea más prioritario que el proceso que va a cargar en memoria principal.

En cuanto a los instantes en los que se activa el proceso swapper hay dos alternativas:

a) Se activa cada vez que la CPU queda ociosa, antes de ejecutar el proceso ocioso.

b) Se activa cada vez que se llama al planificador, antes de que éste seleccione el siguiente proceso que se va a poner en ejecución.

2.1.- Indique qué ventaja tiene la opción "a" respecto a la opción "b".
2.2.- Indique qué contradicción tiene la opción "a" respecto a la política de planificación utilizada (prioridades no expulsora).

3.- Un proceso padre PP ejecuta el siguiente programa ejec1, en el que crea un proceso hijo PH:

```
main() {
  int r, status;

  /* t1 */
  if (fork() != 0){
     r = wait(&status);
  }
  else {
     r = execv("ejec2",0);
  }
}
```

Suponiendo que:

- La memoria principal (de 64 MBytes) en el momento inicial tiene cargados, en este orden:

SO	20 M Bytes
Proceso P1	39 M Bytes
Hueco	512 K Bytes
Proceso PP	3 M Bytes
Proceso P2	1 M Bytes
Hueco	512 K Bytes

- P1 y P2 están listos para ejecutar en memoria, solo consumen CPU y nunca se bloquean.
- Proceso PP está "Ejecutando en usuario", en el instante t1 (indicado en el código en comentario).
- La imagen de memoria del proceso asociado a ejec2 requiere un espacio en memoria de 2 M bytes y solo consume CPU sin bloquearse nunca.
- El orden de prioridades es: prioridad PH < prioridad P2 < prioridad P1 < prioridad PP.
- Con la llamada fork(), si no existe espacio suficiente en memoria para el nuevo proceso, éste se crea y la nueva imagen se lleva al área de swap.
- El proceso swapper, para cargar procesos en la memoria, sigue los criterios del apartado 2.
- No se producen errores en las llamadas al sistema.

Indique qué transiciones realizará el proceso padre PP y cuáles el proceso hijo PH para el caso de la opción "b" del apartado 2 (el proceso swapper se activa cada vez que se llama al planificador). Las transiciones se colocarán de forma cronológica. Explique brevemente el motivo de dichas transiciones.

Apellidos .....

Nombre .....

Nº de Matrícula .....

Nº Orden S.O. I

EJERCICIO 2. Hoja de Respuestas

Apartado 1

Transición que no es posible:

Justificación:

Apartado 2.1

Ventaja:

Apartado 2.2

Contradicción:

Apartado 3

El proceso swapper se activa cada vez que se llama al planificador

Transiciones del proceso **padre (PP)**	Explicación transición de **proceso padre**	Transiciones del proceso **hijo (PH)**	Explicación transición de **proceso hijo**