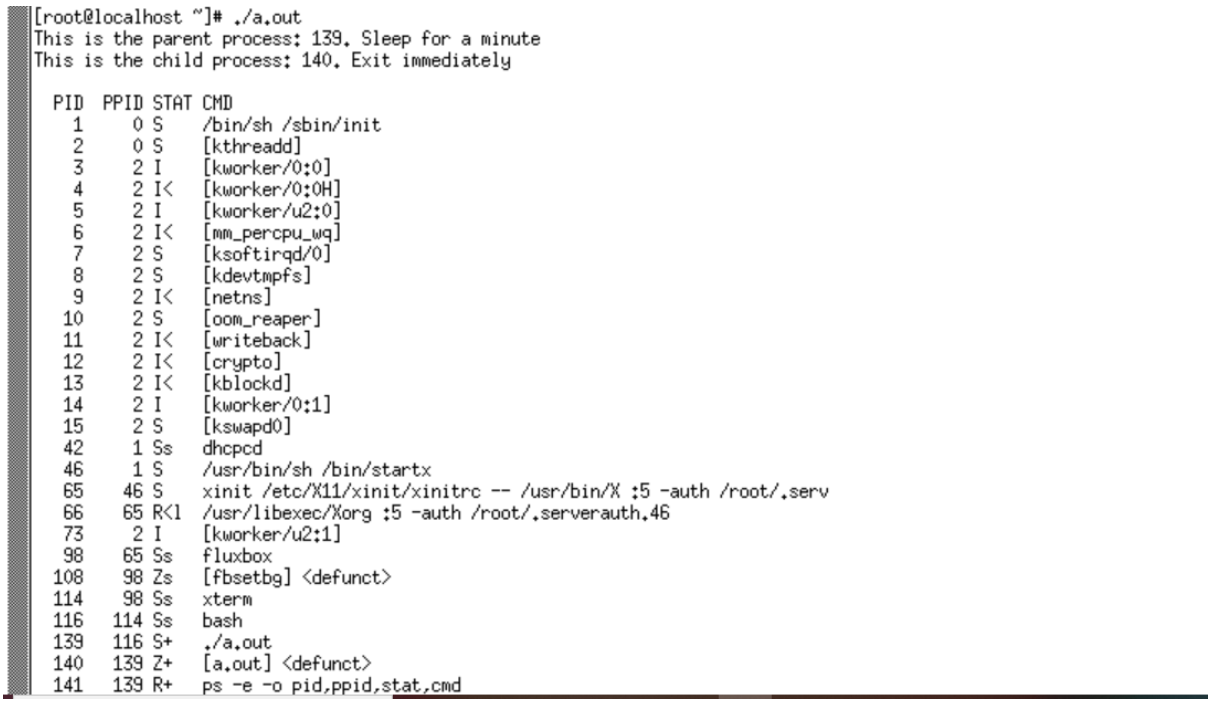
10 A) Write a C/Java program that creates a zombie and then calls system to execute the ps command to verify that the process is zombie.

```
#include<stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
    int main()
    {
        pid_t child_pid;
        child_pid = fork ();
        if (child_pid > 0)
        {
            printf("This is the parent process: %d. Sleep for a minute\n",getpid());
            sleep (60);
        }
        else
        {
            printf("This is the child process: %d. Exit immediately\n",getpid());
            exit (0);
        }
            system("ps -e -o pid,ppid,stat,cmd");
           return 0;
    }
```

Output:

Command to compile : gcc filename.c

Command to run the code : ./a.out



```
[root@localhost ~]# ./a.out
This is the parent process: 139. Sleep for a minute
This is the child process: 140. Exit immediately

  PID  PPID STAT CMD
    1     0 S    /bin/sh /sbin/init
    2     0 S    [kthreadd]
    3     2 I    [kworker/0:0]
    4     2 I<   [kworker/0:0H]
    5     2 I    [kworker/u2:0]
    6     2 I<   [mm_percpu_wq]
    7     2 S    [ksoftirqd/0]
    8     2 S    [kdevtmpfs]
    9     2 I<   [netns]
   10     2 S    [oom_reaper]
   11     2 I<   [writeback]
   12     2 I<   [crypto]
   13     2 I<   [kblockd]
   14     2 I    [kworker/0:1]
   15     2 S    [kswapd0]
   42     1 Ss   dhcpcd
   46     1 S    /usr/bin/sh /bin/startx
   65    46 S    xinit /etc/X11/xinit/xinitrc -- /usr/bin/X :5 -auth /root/.serv
   66    65 R<l  /usr/libexec/Xorg :5 -auth /root/.serverauth.46
   73     2 I    [kworker/u2:1]
   98    65 Ss   fluxbox
  108    98 Zs   [fbsetbg] <defunct>
  114    98 Ss   xterm
  116   114 Ss   bash
  139   116 S+   ./a.out
  140   139 Z+   [a.out] <defunct>
  141   139 R+   ps -e -o pid,ppid,stat,cmd
```

B) Write a C/Java program to avoid zombie process by forking twice.

```
#include<stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

int main()

{

      pid_t  pid;

      if ((pid=fork())< 0)

       {

         printf("Fork  error");

       }

       else if( pid==0)

       {

       printf("first child pid=%d\n", getpid());
```

```
        if((pid=fork())< 0)
        printf("Fork  error");
        else if( pid > 0)
         exit(0);
        sleep(5);
        printf("second child pid = %d\n  parent pid=%d\n", getpid(),  getppid());
        exit (0);
       }
}
```

Output:

Command to compile : gcc filename.c

Command to run the code : ./a.out