Assignment 3 (Due: Thursday, February 7, 11:59pm)

(40pts) 1. Read the following code. Find out how many processes are created when the code is run, and point out the relation (i.e., the parent-child relation) existing among these processes.

```
#include <unistd.h>
#include <stdio.h>

void main(int argc, char *argv[])
{
        int tmp;
        tmp=fork();
        if(tmp>0){
                int tmp;
                tmp=fork();
                if(tmp==0) fork();
        }
        tmp=fork();
        printf("This is process: %d\n", getpid());
}
```

For example, when the following code is executed, two processes are created. Suppose these processes are P0 and P1. P0 is the parent of P1.

```
#include <stdio.h>
#include <unistd.h>
int main( ){
      fork();
}
```

(30pts) 2. Using the program shown below, identify the values of pid, pid1 and value at lines A, B, C, D, E and F. (Note: Function getpid() returns the ID of the calling process. Assume: the IDs of the parent and child processes are 3000 and 5000, respectively.)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main( ){
      pid_t pid, pid1;
      int value;
      pid = fork( );
      value=0;

      if (pid < 0) {
            printf("Fork Failed\n" );
            exit(1);
      }
      else if (pid == 0){
            value+=1;
```

```
            pid1 = getpid( );
            printf("chid: pid = %d\n", pid); /* A */
            printf("child: pid = %d\n", pid1); /* B */
            printf("child: value=%d\n", value); /* C */
      }else{
            value+=2;
            pid1 = getpid( );
            printf("parent: pid = %d\n", pid); /* D */
            printf("parent: pid = %d\n", pid1); /* E */
            wait(NULL);
            printf("parent: value=%d\n", value); /* F */
      }
      return 0;
}
```

(30pts) 3. Read the following program and find out what are the outputs. If there are multiple possibilies, list all the possible outputs.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

void main()
{
        int fd0[2],fd1[2];
        pid_t pid;
        char buf[255];

        pipe(fd0);
        pipe(fd1);
        pid=fork();

        if(pid==0){
                read(fd0[0],buf,255);
                printf("This is child process 1.\n");
                write(fd1[1],"hello",sizeof("hello"));
                exit(0);
        }else{
                pid=fork();
                if(pid==0){
                        printf("This is child process 2.\n");
                        write(fd0[1],"hello",sizeof("hello"));
                        exit(0);
                }
                read(fd1[0],buf,255);
                printf("This is parent process.\n");
        }
}
```