Assignment 2
Due date: Sep. 18th, 23:59

Each question is worth 2 points.

1. How many total processes are created (including the first process running the program)?

```
#include <unistd.h>

int main ()
{
  fork();
  if (fork())
    fork();
  fork();
  return 0;
}
```

2. Explain the circumstances under which the line of code marked printf("how to reach here?\n") will be reached.

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

int main()
{
  pid_t pid;
  pid = fork(); /* fork a child process */
  if (pid < 0) { /* error occurred */
    fprintf(stderr, "Fork Failed");
    return 1;
  } else if (pid == 0) { /* child process */
    execlp("/bin/ls", "ls", NULL);
    printf("how to reach here?\n");
  } else { /* parent process */
    wait(NULL);
    printf("Child Complete\n");
  }
  return 0;
}
```

3. Assume that the actual pids of the parent and child are 25301 and 25302, respectively. Write **one feasible** output result.

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

int main()
{
  pid_t pid, pid1;
  /* fork a child process */
  pid = fork();
  if (pid < 0) { /* error occurred */
    fprintf(stderr, "Fork Failed\n");
    return 1;
  }
  else if (pid == 0) { /* child process */
    pid1 = getpid();
    printf("child: pid = %d\n",pid);
    printf("child: pid1 = %d\n",pid1);
  }
  else { /* parent process */
    pid1 = getpid();
    printf("parent: pid = %d\n",pid);
    printf("parent: pid1 = %d\n",pid1);
    wait(NULL);
  }
  return 0;
}
```

4. **Draw** the process graph for the following code, and write **one feasible** output result.

```
void question4()
{
  printf("L0\n");
  if (fork() == 0) {
    printf("L1\n");
    if (fork() == 0)
      printf("L2\n");
  }
  printf("Bye\n");
```

```
}
```

5. What would be a possible problem if you executed the following program? (You **intend** for it to run forever, so the infinite loop is not the problem!) How can you solve it?

```
int main()
{
  for (;;) {
    if (fork() == 0)
      exit(0);
    sleep(1);
  }
  return 0;
}
```