Lab2

PART A

Exercise 1

可以。

PART B

Exercise 2

`pid = wait((int *) 0);` 的作用是等待进程的退出状态，若将其注释掉，父进程不会等待子进程结束就会输出 `child pid is done`，这样 `child: exiting` 和 `child pid is done` 两行输出顺序会颠倒。

Exercise 3

可以，可以创建子进程时，在子进程中执行 `exec` 系统调用。

```
pid_t pid = fork();
if (pid == 0) {
    ...
    execv("path", argv);
}
```

Exercise 4

```
    char *argv[2];
    argv[0] = "cat";
    argv[1] = 0;
    if(fork() == 0) {
        close(0);
        open("input.txt", O_RDONLY);
        exec("cat", argv);
    }
```

进入子进程后，关闭标准输入文件描述符0，使用 `open` 函数打开文件 `input.txt` ，因为文件描述符0已被关闭，所以该文件将会被分配文件描述符0。这样文件 `input.txt` 的内容就被输入重定向到了子进程的标准输入中。最后使用 `exec` 执行 `cat` 命令。

Exercise 5

顺序很重要，不应该重新排序。
原先的顺序是先关闭标准输入文件描述符0，与最初链接的所有内容断开，再将标准输入重定向到管道的读取端；调整顺序后会先将标准输入连接到管道的读取端，然后关闭文件描述符0，会将标准输入的所有连接断开，实际上与管道也断开了。

Exercise 6

优点有：

1. 写法简便。
2. 不需要创建中间文件。