غلط های موجود در کد زیر را پیدا کرده و نخست علت غلط بودن آنها را بیان کرده و سپس آنها را اصلاح کنید.

```
child_1 = fork();
if (child_1 == 0)
{
    shmID = shmget(key, 2795, 1024, 0666);
    memory = shmat(shmID, null, 0);
    child_2 = fork();
    if (child_2 == 0)
    {
        shmID = shmget(key, 2395, 1024, 0666);
        memory2 = shmat(shmID, value, 0);
        write_to_memory(memory2, "child_2 message");
    }
    else
    {
        text = read_from_memory(memory);
        if (text == "kill child_2") {
            kill(child_2);
            write_to_memory(memory, "killed child_2");
        }
        else
        {
            shmID2 = shmget(key, 2395, 1024, 0666 | IPC_CREAT);
            memory2 = shmat(shmID2, value, 0);
            text = read_from_memory(memory2);
            print("child_2 wrote:", text);
            text = read_from_memory(memory);
            print("child_1 wrote:", text);
            write_to_memory(memory, "kill child_2");
            text = read_from_memory(memory);
            print("child_1 wrote:", text);
        }
    }
}
```

- اولین مشکل در استفاده از تابع shmget است. این تابع تنها ۳ ورودی دارد که اولین ورودی آن، کلید حافظه اشتراکی است، سپس اندازه این حافظه و پس از آن permission آخرین ورودی این تابع است. برای اصلاح این مشکل باید key را حذف کنیم و از و از خود کلیدها که پس از آن و به صورت رشته عددی آمده‌اند استفاده کنیم.
- در استفاده از تابع shmat برای گرفتن حاضه اشتراکی memory2، به ورودی دوم، متغیر value داده شده است که یک متغیر تعریف نشده است. ورودی دوم این تابع آدرس حافظه‌ای را میگیرد که فرآیند مربوط به حافظه اشتراکی attach میشود. برای اصلاح میتوانیم از null به جای value استفاده کنیم.
- اگر child_2 زودتر از child_1 اجرا شود برنامه هنگام اجرای shmget با خطا روبه‌رو میشود چرا که حافظه‌ای اشتراکی memory2 ساخته نشده است پس نمیتوان به آن دسترسی پیدا کرد. برای حل این مشکل کافی است به جای 666 قرار دهیم 666 | IPC_CREATE.

- فرآیند پدر زودتر از child_1 تمام میشود بنابراین این فرآیند یتیم میشود که این موضوع میتواند باعث ایجاد مشکل در اجرای این فرآیند شود به همین دلیل از wait استفاده میکنیم. در childe_1 نیز باید از wait استفاده کنیم.

پس از اصلاح مشکلاتی که در بالا ذکر شد، برنامه به صورت زیر خواهد بود:

```
int main()
{
    pid_t child_1 = fork();
    if (child_1 == 0)
    {
        int shmID = shmget(2795, 1024, IPC_CREAT | 0666);
        void *memory = shmat(shmID, NULL, 0);
        pid_t child_2 = fork();
        if (child_2 == 0)
        {
            int shmID2 = shmget(2395, 1024, IPC_CREAT | 0666);
            void *memory2 = shmat(shmID2, NULL, 0);
            write_to_memory(memory2, "child_2 message");
        }
        else
        {
            char *text = read_from_memory(memory);
            if (strcmp(text, "kill child_2") == 0)
            {
                kill(child_2, SIGKILL);
                write_to_memory(memory, "killed child_2");
            }
            else
            {
                int shmID2 = shmget(2395, 1024, IPC_CREAT | 0666);
                void *memory2 = shmat(shmID2, NULL, 0);
                text = read_from_memory(memory2);
                printf("child_2 wrote: %s\n", text);
                text = read_from_memory(memory);
                printf("child_1 wrote: %s\n", text);
                write_to_memory(memory, "kill child_2");
                char *text = read_from_memory(memory);
                printf("child_1 wrote: %s\n", text);
                shmdt(memory2);
            }
            wait(NULL);
            shmdt(memory);
        }
    }
    else
    {
        wait(NULL);
    }
    return 0;
```

}