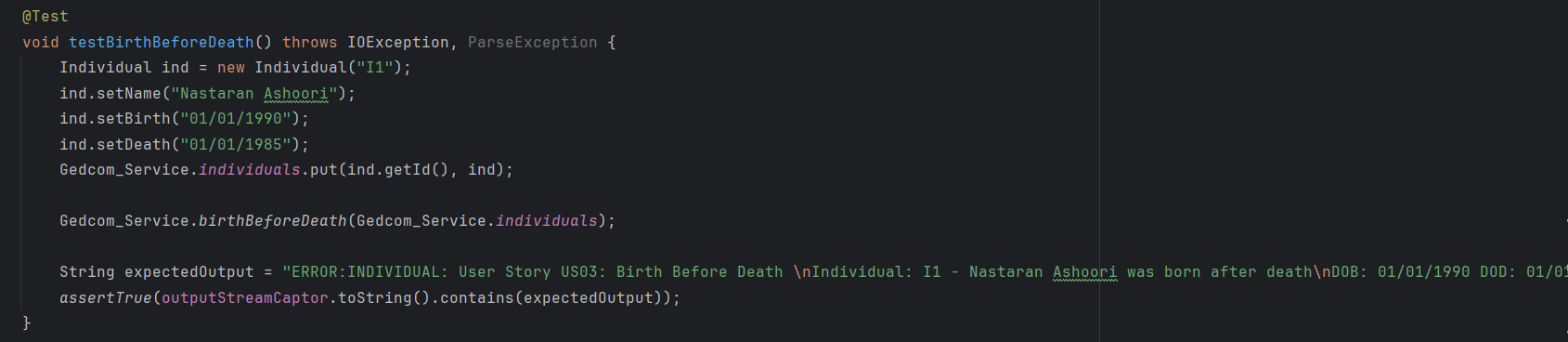
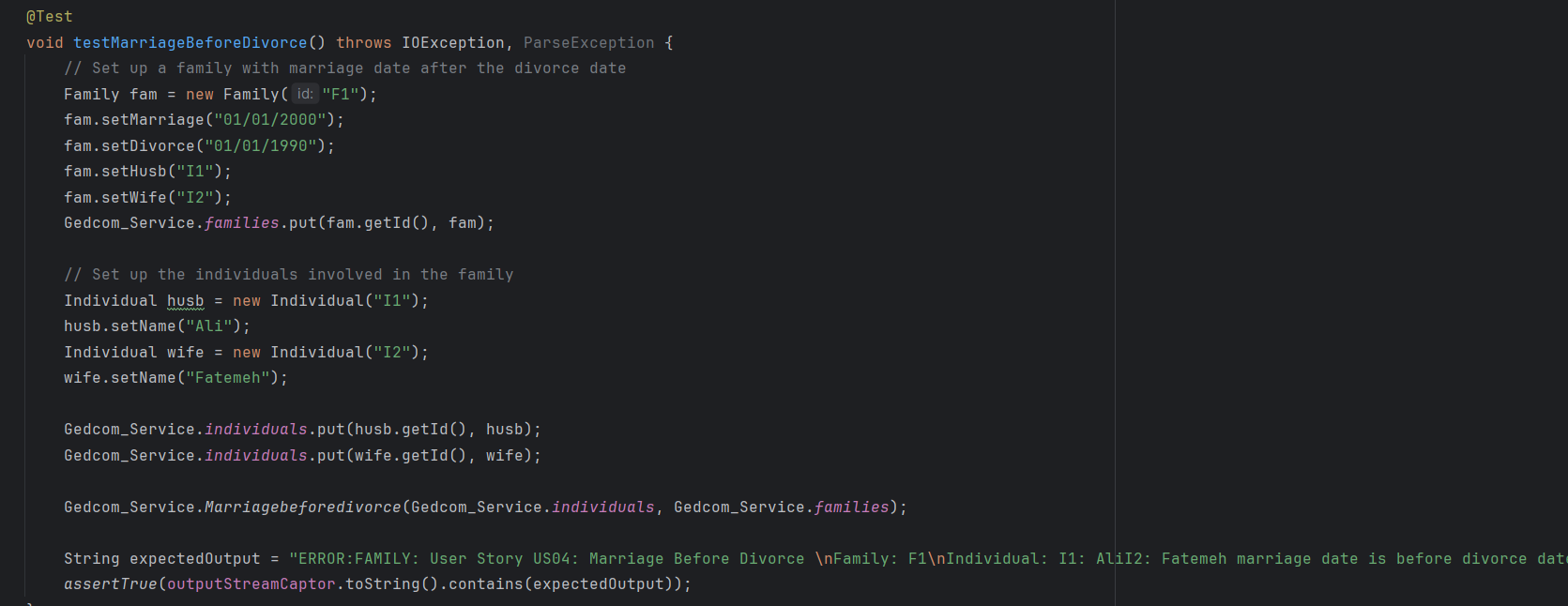
برای اجرا شدن ابتدا فایل pom.xml را تغییر میدهیم. سپس برای کلاس Gedcom\_ServiceTest تست های زیر را مینویسم.

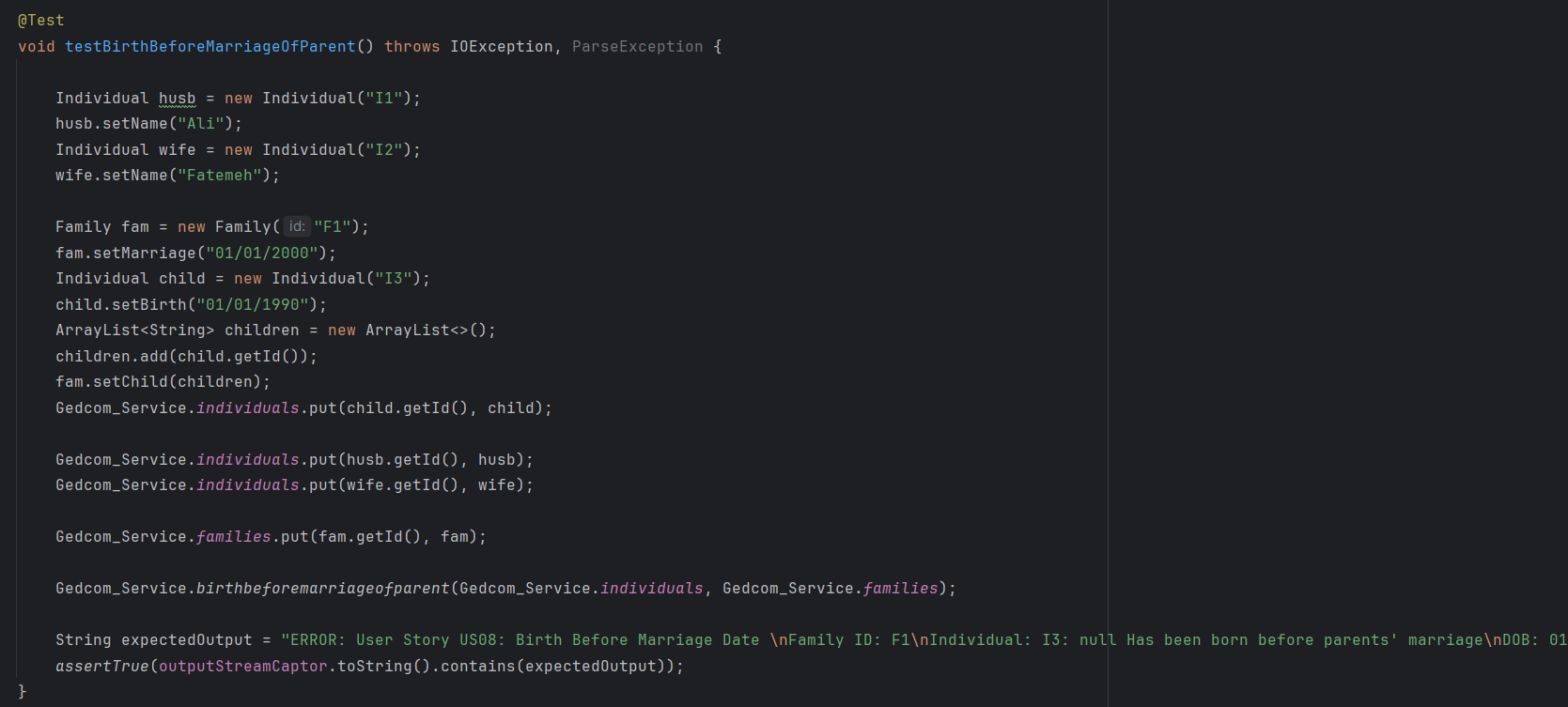
1. testBirthBeforeDeath : این تابع چک میکند که اگر در سیستم شخصی ثبت شود که تولدش بعد از مرگش باشد، سیستم حتما پیغام خطا داده و آن را در فایل چاپ کند.



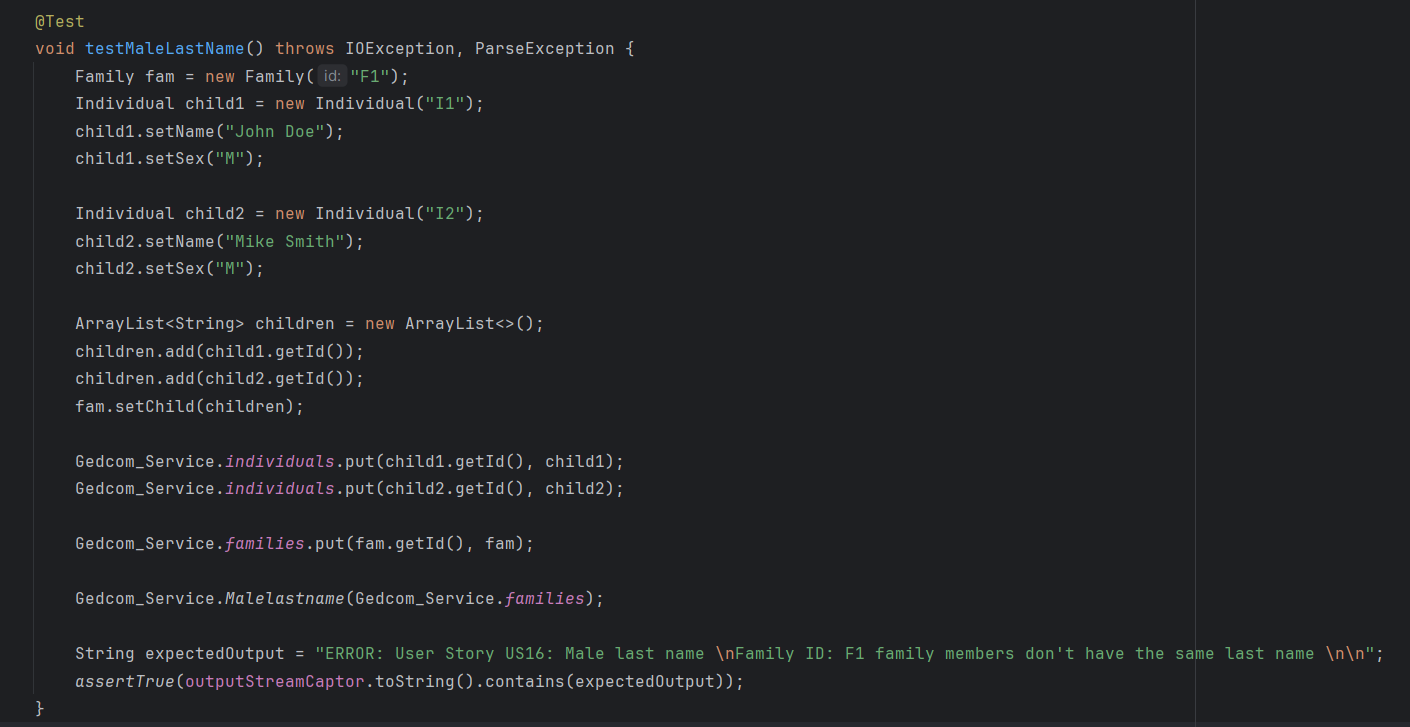
1. testMarriageBeforeDivorce: این تابع چک میکند که اگر تاریخ طلاق 2 شخص قبل از تاریخ ازدواجشان باشد سیستم حتما پیغام خطا داده و آن را در فایل چاپ کند.



1. testBirthBeforeMarriageOfParent: این تابع چک میکند اگر تاریخ تولد فرزندی قبل از ازدواج پدر مادرش باشد ، سیستم حتما پیغام خطا داده و آن را در فایل چاپ کند.



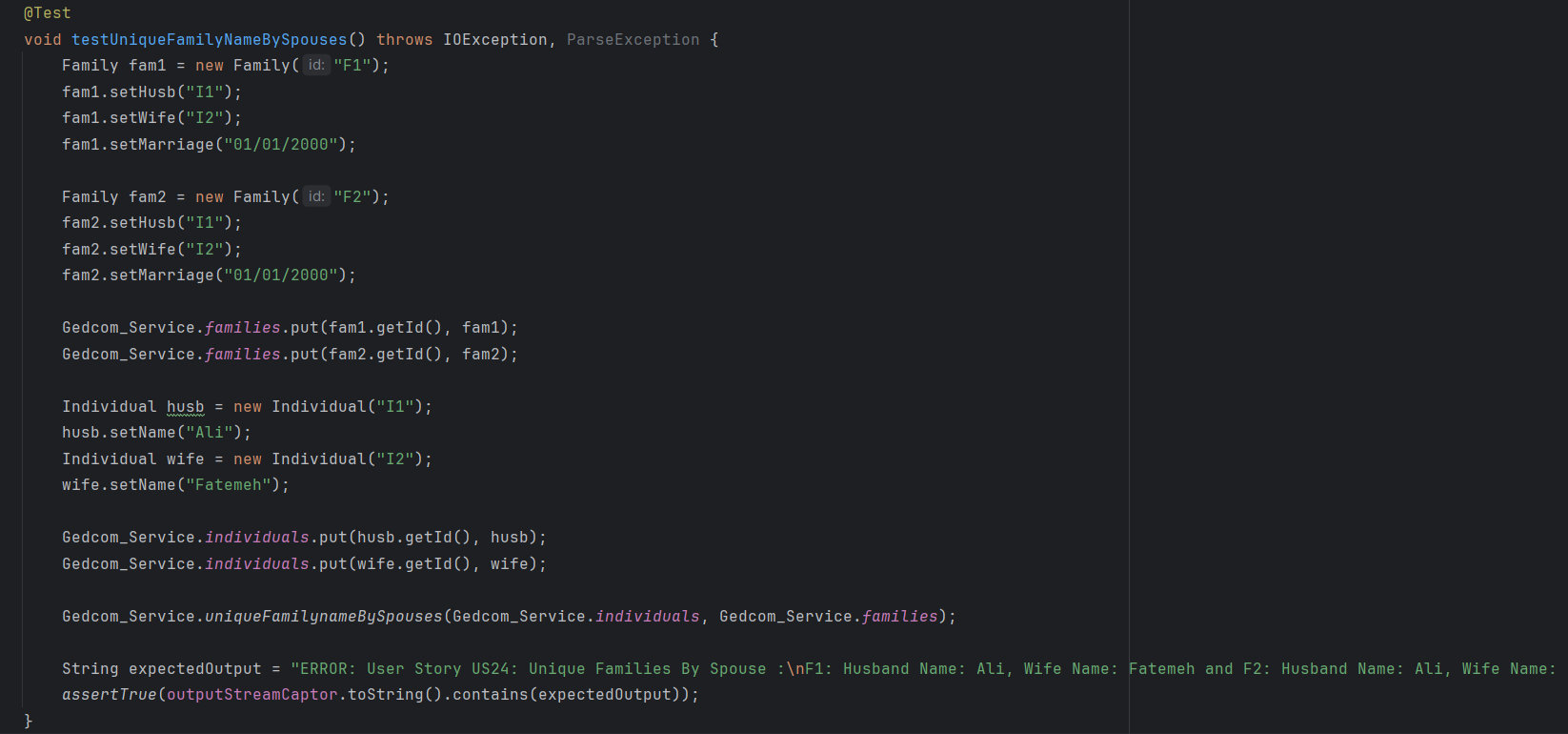
1. testMaleLastName: این تابع چک میکند نام خانوادگی تمامی مردان یک خانواده یکسان باشد و در غیر این صورت سیستم حتما پیغام خطا داده و آن را در فایل چاپ کند.



1. testAuntsAndUnclesName: این تابع چک میکند ازدواجی بین فرزندان یک خانواده و عمه یا عموهایشان صورت نگرفته باشد و در غیر این صورت سیستم حتما پیغام خطا داده و آن را در فایل چاپ کند.

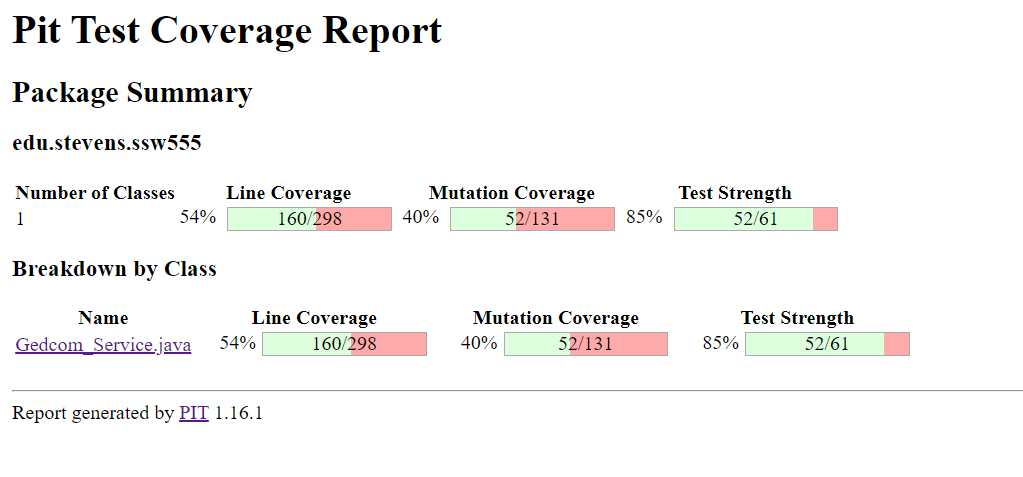


1. testUniqueFamilyNameBySpouses : این تابع چک میکند که خانواده های تکراری ثبت نشده باشد و در غیر این صورت سیستم حتما پیغام خطا داده و آن را در فایل چاپ کند.(خانواده تکراری یعنی 2 شخص یکسان در یک روز یکسان ازدواج کرده باشند اما 2 رکورد برایشان ثبت شده باشد.)



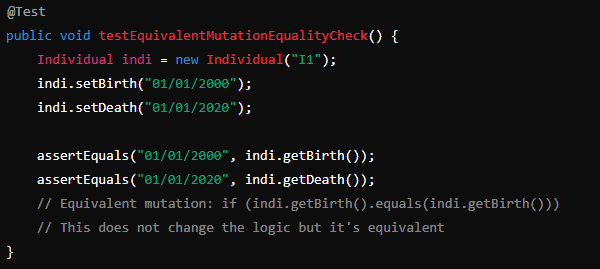
1. هم چنین دو تست نیز برای چک کردن ایجاد فایل خروجی نا معتبر و خواندن فایل نامعتبر ایجاد شده است.

در نهایت pitest را اجرا میکنیم. نتیجه کلی به شرح زیر است.



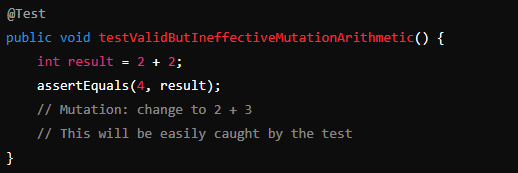
فایل Gedcom\_serviceTest.java.html ایجاد شده است که گزارش کاملی از mutation ها ارائه میکند. حال به تحلیل موارد خواسته شده میپردازیم.

1. جهش های نا معتبر: این جهش مثلا باعث ایجاد ارور syntax میشود یا یک فراخوانی تابع را انجام نمیدهد. از این نوع جهش نداشته ایم.
2. معادل کد اصلی : تست زیر نمونه ای از جهش های معادل کد اصلی است. اگر مثلا به جای assert equal از توابع مشابه برای سنجش تساوی استفاده کنیم.



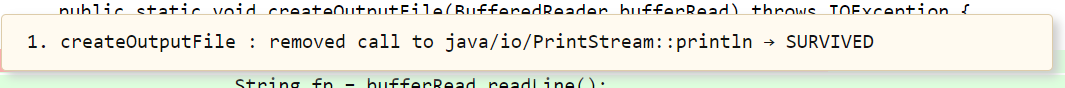
از این نوع جهش در نتیجه pitest ایجاد نشده است. در واقع این نوع جهش ها در کد ما کاور نشده است.

1. جهش های معتبر ولی نا مفید که توسط اکثر تست ها قابل تشخیص است.



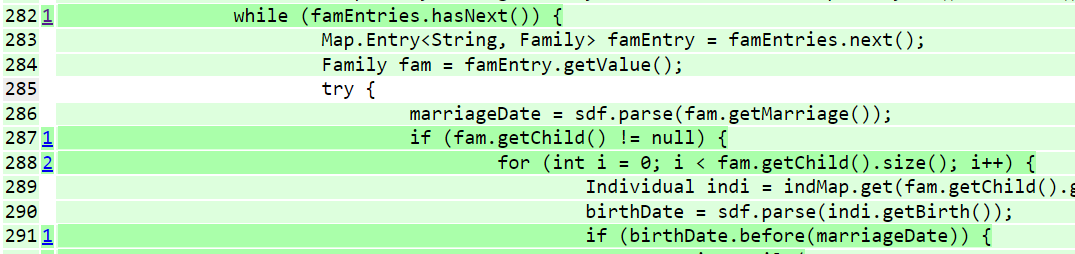
مثلا در جهش های ایجاد شده جهش زیر را داریم که معتبر ولی نامفید است. این جهش ها منطق کلی برنامه را زیر سوال نمیبرند. ولی به راحتی با اعمال این جهش ها ( مثلا حذف print) تست fail میشود و توسط هر تستی میتوان آن را به دست آورد.

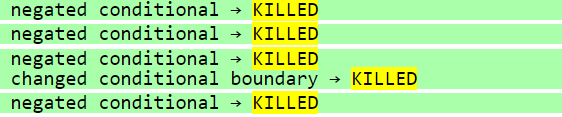




تغییر در توابع پرینت ارورهای نهایی نیز جزو همین دسته است. این جهش ها kill میشود ولی مفید نیستند.

1. جهش های مفید: این جهش ها killed شده هستند ولی تعداد کمی از تست ها آن را تشخیص میدهند. جهش ایجاد شده مفید در واقع نقاط کلیدی تابع را تست میکند و با تغییر آن منطق کلی برنامه زیر سوال میرود. نقیض کردن شروط از این نوع جهش هاست.



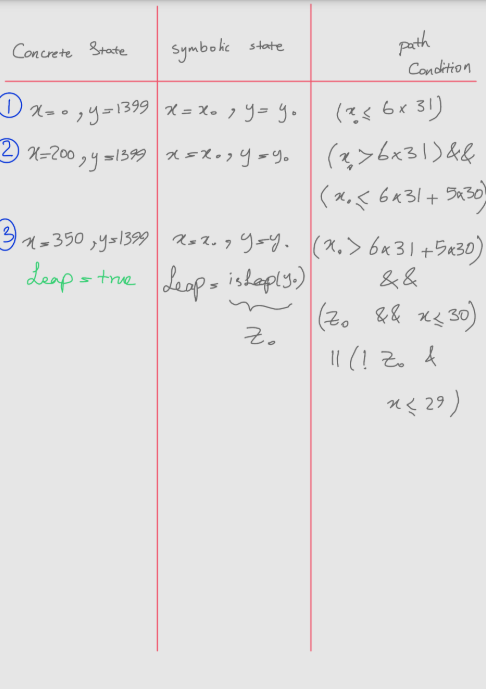


1. جهش های زنده

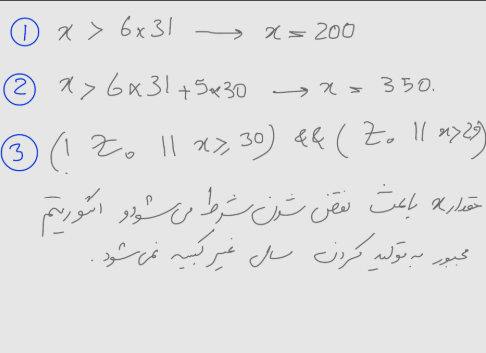
جهش هایی که در رده survive قرار دارند از این دسته هستند. در اینجا جهش صورت گرفته ولی تست ما متوجه تغییر نشده و fail نشده است.

1. جهش های تکراری: از این نوع جهش در جهش ها نداشتیم.

سوال 2) الف)



توضیحات نقیض کردن هر مرحله در تصویر بعد آمده است.



در نهایت برای catch کردن ارور باید مقدار 370 را به عنوان x بدهیم.

ب)

