تمرین چهارم درس مهندسی نرم‌افزار

گروه پنجم

اعضای گروه:

امیرحسین فراهانی 97106154
سارا آذرنوش 98170668
پیمان حاجی محمد 98170776
محمدعلی حسین نژاد 98170787
رویا قوامی 98171031

1.

- bug: یا بعبارتی گیر و اشکال در برنامه نویسی که باعث ایجاد مشکلاتی اعم از کرش کردن برنامه، گیر کردن در حلقه بینهایت، تولید خروجی نادرست و... می شوند. از جمله علت بروز آنها میتوان به اشتباهات کد نویسی اشاره کرد
- issue: یا همان مسئله، ممکن است در طول فرآیند توسعه پیش بیاید. این مسئله ممکن است مربوط به بازخورد کاربر، مشکلات سیستمی یا نیاز به تغییرات در نرم افزار باشد. برای حل مسائل، نیاز به بررسی دقیق و پیدا کردن یک راه حل مناسب داریم.
- defect: یا همان عیب و نقص، به تفاوت بین خروجی مد نظر با خروجی واقعی گفته می شود. عیب ها را تستر ها در فاز تست شناسایی می کنند، همچنین توسعه دهندگان نیز می توانند در فاز توسعه نیز آنها را شناسایی کرده و برطرف کنند.
- Fault: یا همان خطا، یک گام اشتباه، فرآیند اشتباه یا تعریف داده ای اشتباه می باشد که سیستم را به حالتی می برد که قرار نبود باشد و پیش بینی نشده بود در آن حالت باشد. اشتباهات انسانی باعث بروز این خطا می شود.
- error: یا بعبارتی اشتباه، یک اشتباه، سو برداشت یا سو تفاهم در انجام فرآیند توسعه نرم افزار است، برای مثال برنامه نویس، برداشتی اشتباه از طراحی داشته، برنامه نویس در روند برنامه نویسی نام یک متغیر را اشتباه تایپ کرده. معمولا error ها باعث عدم امکان اجرای درست و صحیح برنامه و یا حتی کامپایل آنها می شوند

- failure: شکست، زمانی اتفاق می افتد که نرم افزار، نیازمندی هایی که برایش تعریف شده را نتواند به درستی برطرف کند و زمانی که به کاربر نهایی می رسد با مشکلاتی مواجه شود

.2

- object oriented testing یک روش آزمون نرم افزار است که بر مبنای اصول شی گرایی و طراحی شی گرا انجام می شود. در این روش، نرم افزار به عنوان مجموعه ای از شی ها در نظر گرفته می شود و هر شی به صورت جداگانه آزمون می شود. پنج مورد از انواع این نوع تست را در ادامه توضیح خواهیم داد:
- unit testing: در این روش، هر شی به صورت جداگانه توسط تست های واحد آزموده می شوند. در این روش کوچک ترین واحد برای تست کردن، کلاس کپسوله شده می باشد
- integration testing: این نوع تست، تعامل بین شی های مختلف در سیستم را مورد آزمایش قرار می دهد
- system testing: در این مدل از آزمون نحوه عملکرد کل سیستم به عنوان یک موجودیت کل با تست های سطح بالا مورد بررسی قرار می گیرد
- thread base testing: در این روش تمامی کلاس های لازم برای تحقق یک use case مشخص در کنار هم قرار گرفته و تست می شوند
- use base testing: رابط ها و سرویس های ماژول ها به صورت سطح به سطح در سراسر سلسله مراتب آزمایش می شوند. در ابتدا کلاس های مستقل که استفاده بسیار کمی از کلاس های سرور دارند آزمایش می شوند. سپس سطح بعدی کلاس ها یعنی کلاس های وابسته مورد آزمایش قرار می گیرند، فرآیند تا زمانی ادامه دارد که کل سیستم آزموده شده باشد
- خیر، آزمایش هر webapp ای را نمی توان مورد آزمون قرار داد زیرا پیچیدگی و دامنه وب اپ ها می‌تواند بسیار متفاوت باشد و آزمایش هر سناریو یا ترکیبی از ورودی‌ها ممکن است عملی نباشد. علاوه بر این، برخی از برنامه های وب ممکن است محدودیت های امنیتی یا مسائل مربوط به حفظ حریم خصوصی داشته باشند که انواع خاصی از آزمایش را غیر ممکن یا غیر عملی می کند.

.3

1- اعتبارسنجی فرآیندی است برای اطمینان از اینکه یک سیستم، هدف مورد نظر خود را برآورده می کند و نیازهای کاربران خود را نیز برآورده می کند. به طور کلی می توان مراحل انجام تست اعتبارسنجی را به شکل زیر بیان کرد:

- تعریف معیارهای اعتبارسنجی

اولین گام، شامل تعریف معیارهایی است که سیستم یا محصول باید برای اینکه معتبر تلقی شود آنها را داشته باشد. این معیارها ممکن است شامل functional requirementها، performance requirementها و سایر معیارها باشد.

- برنامه ریزی آزمایش

پس از تعریف معیارهای اعتبار سنجی، یک نقشه آزمایش(test plan) باید ایجاد شود که رویکرد آزمایش، موارد آزمایش و نتایج مورد انتظار را مشخص کند.

- اجرای آزمون ها

آزمون ها باید طبق نقشه آزمایش اجرا شوند که ممکن است شامل تست دستی، تست خودکار یا ترکیبی از هر دو باشد.

- تجزیه و تحلیل نتایج

پس از تکمیل آزمایشات، نتایج باید تجزیه و تحلیل شوند تا مشخص شود که آیا سیستم یا محصول دارای معیارهای اعتبارسنجی است یا خیر. هر گونه نقص یا مشکل باید ثبت و اولویت بندی شود.

- رفع مشکلات

هر گونه defect یا مشکلی که در طول تست اعتبار سنجی شناسایی می شود، باید با تصحیح کد یا تغییر requirementها برطرف شود.

- تکرار آزمایش

پس از رفع هر گونه مشکل، سیستم یا محصول باید مجددا آزمایش شود تا اطمینان حاصل شود که معیارهای اعتبارسنجی را برآورده می کند.

- مستندسازی نتایج

در نهایت، نتایج آزمایش اعتبارسنجی باید مستند شود، که این مستند سازی شامل هر گونه نقص پیدا شده، نحوه رفع آنها و اینکه آیا سیستم یا محصول دارای معیارهای اعتبارسنجی است یا خیر می باشد.

2- بله، نظر کاربر (user) جنبه مهمی از تست اعتبار سنجی است. زیرا می تواند بینش ارزشمندی در مورد usability و عملکرد سیستم ارائه دهد. تست پذیرش کاربر (UAT یا User Acceptance Testing) نوعی تست اعتبار سنجی است که شامل آزمایش کاربران نهایی سیستم در یک محیط واقعی است تا اطمینان حاصل شود که نیازها و انتظارات کاربران برآورده می شود. از بازخورد UAT می توان برای شناسایی هر گونه مشکل یا زمینه هایی که قبل از انتشار محصول می توان نرم افزار را در آنها بهبود بخشید استفاده کرد.

3- تست اعتبار سنجی (validation testing) و تست تایید (verification testing) دو نوع مختلف تست نرم افزار هستند. تست تأیید، فرآیند آزمایش یک محصول یا سیستم نرم افزاری است که برای اطمینان از مطابقت با مشخصات و الزامات (requirement های) طراحی آن انجام می شود. این تست برای اطمینان یافتن از این است که نرم افزار به درستی ساخته شده است و با مشخصات طراحی محصول مطابقت دارد. تست تایید معمولا شامل فعالیت هایی مانند تست واحد (unit testing)، تست یکپارچه سازی (integration testing) و تست سیستم (system testing) است.

از سوی دیگر، تست اعتبارسنجی، فرآیند آزمایش یک محصول یا سیستم نرم افزاری است تا اطمینان حاصل شود که نیازهای کاربران و ذینفعان خود را برآورده می کند. تست اعتبارسنجی برای اطمینان از این موضوع است که نرم افزار کار درستی انجام می دهد و نیازها و انتظارات کاربران مورد نظر خود را برآورده می کند. تست اعتبار سنجی معمولا شامل فعالیت هایی مانند تست پذیرش کاربر (user acceptance testing)، تست قابلیت استفاده (usability testing) و تست اکتشافی (exploratory testing) است.

آزمایش تأیید به طور معمول در طول فرآیند توسعه انجام می شود، در حالی که آزمایش اعتبارسنجی معمولاً پس از تکمیل فرآیند توسعه و آماده شدن نرم افزار برای انتشار انجام می شود.

4- تست سازگاری (Compatibility testing) نوعی تست نرم افزاری است که سازگاری یک برنامه نرم افزاری را با سخت افزار، نرم افزار، سیستم عامل، مرورگرها و محیط های مختلف شبکه ارزیابی می کند. هدف از تست سازگاری این است که اطمینان حاصل شود که برنامه نرم افزاری می

تواند به درستی و یکپارچه در پلتفرم ها و پیکربندی های مختلف عمل کند. این نوع آزمایش برای اطمینان از اینکه برنامه نرم افزاری می تواند توسط طیف گسترده ای از کاربران با دستگاه ها و تنظیمات مختلف بدون هیچ گونه مشکل سازگاری استفاده شود، مهم است. تست سازگاری را می توان به صورت دستی یا از طریق ابزارهای تست خودکار انجام داد. برخی از انواع رایج تست سازگاری عبارتند از:

- **تست سازگاری مرورگر:** این تست شامل آزمایش نرم افزار بر روی مرورگرهای مختلف وب مانند کروم، فایرفاکس، سافاری و اینترنت اکسپلورر است تا از درست کارکردن و نمایش صحیح آن اطمینان حاصل شود.
- **تست سازگاری سیستم عامل**: این تست شامل تست نرم افزار بر روی سیستم عامل های مختلف مانند ویندوز، مک او اس و لینوکس است تا مطمئن شوید که درست کار می کند و پایدار است.
- **تست سازگاری دستگاه تلفن همراه:** این تست شامل تست نرم‌افزار بر روی دستگاه‌ها و پلتفرم‌های تلفن همراه مختلف، مانند iOS و Android، برای اطمینان از عملکرد صحیح و بهینه‌سازی آن برای استفاده از تلفن همراه است.

4.

1- برای یک سیستم بزرگ و پیچیده مانند دیجی کالا، باید از یک رویکرد جامع تست استفاده شود تا اطمینان حاصل شود که سیستم قابل اعتماد، کاربردی و مورد پسند کاربر است. در اینجا چند استراتژی و رویکرد آزمایشی وجود دارد که می تواند برای آزمایش دیجی کالا استفاده شود:

تست عملکردی (Functional Testing): که شامل آزمایش سیستم است تا از اینکه همه ویژگی ها و عملکردهای آن به درستی کار می کنند، اطمینان حاصل شود. این تست می تواند شامل آزمایش عملکرد جستجوی محصول، لیست محصولات و صفحات جزئیات، عملکرد سبد خرید، فرآیند پرداخت، پردازش پرداخت و سایر ویژگی های حیاتی سیستم باشد.

تست پذیرش کاربر: این تست شامل آزمایش سیستم با کاربران واقعی است تا اطمینان حاصل شود که استفاده از آن آسان است و نیازهای کاربران را برآورده می کند. تست پذیرش کاربر می‌تواند شامل تست قابلیت استفاده، نظرسنجی کاربران و روش‌های دیگر برای جمع‌آوری بازخورد از کاربران باشد.

تست عملکرد: این تست شامل آزمایش سیستم برای اطمینان از اینکه می تواند حجم زیادی از ترافیک و تراکنش ها را بدون کاهش سرعت یا خرابی انجام دهد، انجام می شود. تست عملکرد می

تواند شامل تست بار (load testing)، تست استرس و روش های دیگر برای شبیه سازی حجم بالای ترافیک و تراکنش باشد.

تست امنیتی: این تست شامل آزمایش سیستم برای اطمینان از ایمن بودن و محافظت در برابر تهدیدات امنیتی بالقوه مانند هک، نقض داده ها یا دسترسی غیرمجاز است. تست امنیتی می تواند شامل تست نفوذ، تست آسیب پذیری و روش های دیگر برای شناسایی و رفع آسیب پذیری های امنیتی بالقوه باشد.

تست سازگاری: این تست شامل آزمایش سیستم برای اطمینان از عملکرد صحیح و پایدار بودن آن در انواع پلتفرم‌ها، دستگاه‌ها و مرورگرهای وب است. تست سازگاری می تواند شامل آزمایش بر روی سیستم عامل های مختلف، مرورگرهای وب و دستگاه های تلفن همراه باشد.

تست محلی سازی (localization): این تست شامل آزمایش سیستم برای اطمینان از عملکرد کامل آن و ارائه یک تجربه کاربری خوب در زبان ها و مناطق مختلف است که می‌تواند شامل آزمایش سیستم با کاربرانی باشد که به زبان‌های مختلف صحبت می‌کنند، تأیید اینکه سیستم زبان و ارز (currency) صحیح را نشان می‌دهد و اطمینان از اینکه سیستم با مقررات محلی مطابقت دارد.

تست رگرسیون (regression): این تست شامل آزمایش سیستم پس از انجام تغییرات است تا اطمینان حاصل شود که تغییرات هیچ باگ یا مشکل جدیدی ایجاد نکرده است. تست رگرسیون را می توان با استفاده از ابزارهایی مانند سلنیوم (Selenium) یا Cypress به صورت خودکار انجام داد.

2- در مورد سیستم دیجی کالا، باید به دنبال خطاهای مختلفی باشیم که می تواند تجربه کاربری و عملکرد سیستم را تحت تاثیر قرار دهد. در اینجا برخی از انواع خطاها وجود دارد که ممکن است رخ دهد:

خطاهای عملکردی: خطاهایی هستند که سیستم را از انجام صحیح وظایف مورد نظر خود باز می دارند. به عنوان مثال، زمانی که عملکرد جستجوی محصول نتایج دقیقی را برنمی‌گرداند، یا زمانی که فرآیند پرداخت به درستی تکمیل نشده باشد.

خطاهای رابط کاربری (UI): این ها خطاهایی هستند که ظاهر یا قابلیت استفاده رابط کاربری سیستم را تحت تأثیر قرار می دهند. به عنوان مثال، اگر دکمه ها یا پیوندها به درستی کار نکنند یا اگر طرح بندی صفحات به درستی در دستگاه ها یا مرورگرهای وب مختلف نمایش داده نشود.

خطاهای عملکرد: اینها خطاهایی هستند که بر سرعت یا عملکرد سیستم تأثیر می گذارند، مانند زمان بارگذاری کند یا صفحات بدون پاسخ.

خطاهای امنیتی: اینها خطاهایی هستند که امنیت سیستم را تحت تأثیر قرار می دهند، مانند آسیب پذیری هایی که می توانند توسط هکرها مورد سوء استفاده قرار گیرند یا دسترسی غیرمجاز به داده های کاربر.

خطاهای سازگاری: خطاهایی هستند که بر سازگاری سیستم با پیکربندی های مختلف سخت افزار، نرم افزار یا مرورگر تأثیر می گذارند. به عنوان مثال، اگر سیستم در نسخه خاصی از یک مرورگر وب یا در یک دستگاه تلفن همراه خاص به درستی کار نمی کند.

خطاهای محلی سازی: خطاهایی هستند که بر عملکرد یا قابلیت استفاده سیستم در زبان ها یا مناطق مختلف تأثیر می گذارند. به عنوان مثال، اگر سیستم زبان یا واحد پولی را برای یک منطقه خاص نمایش ندهد یا با مقررات محلی مطابقت نداشته باشد.

خطاهای داده: خطاهایی هستند که بر صحت یا یکپارچگی داده ها در سیستم تأثیر می گذارند. به عنوان مثال، اگر اطلاعات محصول گم شده یا نادرست است، یا اگر اطلاعات کاربر به درستی ذخیره یا پردازش نشده باشد.

-3

سناریوی 1: کاربر محصولی را جستجو می کند و آن را به سبد خرید خود اضافه می کند.

رویکرد تست:

تست عملکردی: بررسی می کنیم که عملکرد جستجوی محصول نتایج دقیقی را بر اساس درخواست جستجوی کاربر برمی‌گرداند. همچنین بررسی می کنیم که اطلاعات محصول نمایش داده شده در صفحه نتایج جستجو و صفحه جزئیات محصول صحیح باشد.

تست رابط کاربری: بررسی کنید که صفحات جستجوی محصول و جزئیات محصول به درستی در دستگاه‌ها و مرورگرهای وب مختلف نمایش داده می‌شوند. همچنین بررسی کنید که دکمه ها و پیوندها به درستی کار می کنند، مانند دکمه "افزودن به سبد خرید".

تست عملکرد: بررسی می کنیم که صفحات جستجوی محصول و جزئیات محصول به سرعت بارگیری می شوند و سیستم می تواند حجم زیادی از جستجوها و بازدیدهای محصول را انجام دهد.

تست امنیتی: بررسی میکنیم که داده‌های کاربر به صورت ایمن ذخیره می‌شوند و هیچ آسیب‌پذیری در فرآیند جستجو یا پرداخت محصول وجود ندارد که توسط هکرها مورد سوء استفاده قرار گیرد.

سناریوی 2: کاربر با استفاده از فرآیند پرداخت خرید را تکمیل می کند.

رویکرد تست:

تست عملکردی: بررسی می کنیم که فرآیند پرداخت به درستی کار می کند و کاربر می تواند اطلاعات حمل و نقل و پرداخت خود را به درستی وارد کند. همچنین، بررسی می کنیم که سفارش به درستی پردازش شده است و کاربر یک ایمیل تأیید دریافت می کند.

تست رابط کاربری: بررسی می کنیم که فرآیند پرداخت به درستی در دستگاه‌ها و مرورگرهای وب مختلف نمایش داده می‌شود. همچنین بررسی می کنیم که دکمه‌ها و پیوندها به درستی کار می‌کنند، مانند دکمه «سفارش دادن».

تست عملکرد: بررسی می کنیم که فرآیند پرداخت سریع و پاسخگو باشد و سیستم بتواند حجم زیادی از سفارشات و تراکنش ها را انجام دهد.

تست امنیتی: بررسی می کنیم که فرآیند تسویه حساب ایمن است و داده های کاربر به طور ایمن ذخیره و پردازش می شوند. همچنین، بررسی می کنیم که هیچ آسیب‌پذیری در فرآیند پرداخت وجود نداشته باشد که توسط هکرها مورد سوء استفاده قرار گیرد.

5.

الف) به صورت کلی ما در فرآیند مهندسی نرم افزار میگوییم که باید در سریعترین زمان ممکن ریلیزهای جدید داشته باشیم. اما این موضوع ممکن است به پادالگو تبدیل شود. چرا؟ کاربر در فضایی حرکت میکند که هر لحظه ممکن است با باگ‌ها و خطاهای زیادی مواجه شود درست مثل میدان مین که این خود موجب از بین رفتن اعتماد کاربران به سیستم ما خواهد شد.
حالا چگونه با این مشکل مواجه شویم؟ با ایجاد فرآیندهای تست در سازمان (امروزه وجود پوزیشن هایی مانند qa و qc در تیم‌های بزرگ به همین خاطر است) ما میتوانیم قبل از ریلیز از مشکلات آگاه شویم. سکس

از راه‌ها نوشتن تست کیس و سناریوهای مختلف تست و چک کردن دستی است و حرکت دیگر ایجاد فرآیندهای تست خودکار است (مثل unit test)

ب) به مجموعه‌ای از تست کیس ها test suite میگویند. در واقع به صورت منطقی تست کیس ها رو گروه‌بندی می‌کند. مثلا شما فرض کنید تعدادی تست کیس برای فرآیند لاگین وبسایت دارید به این مجموعه از تست کیس ها، test suite لاگین میگویند. به وجود اوردن این مفهوم به ما کمک می کند در هر بار تغییر مربوط به لاگین دسته مربوطه را اجرا کنیم و چیزی را جا نندازیم.

ج) تست نرم افزار روشی است برای بررسی اینکه آیا محصول نهایی نرم افزار با الزامات مورد انتظار مطابقت دارد و یا اینکه اطمینان حاصل شود که محصول نرم افزار باگی ندارد. در تست اتوماتیک این تست ها عامل انسانی ندارند و توسط کامپیوتر اجرا می شوند. به صورت کلی هدف از تست نرم افزار شناسایی خطاها و یا سنجش درست عملکرد سیستم و تطابق آن با اهداف از پیش تعیین شده است.

این تست های خودکار انواع مختلفی از تست ها را مثل تست امنیتی، تست کاربردپذیری، تست های فانکسنال و ... شامل می‌شوند.

یکی از ابزارهای مورد استفاده برای استفاده از تست های اتوماتیک نرم افزارها selenium است. این ابزار به ما کمک می کند اسکریپت‌هایی به زبان های مختلف برای تست نرم افزارهای تحت وب بنویسیم. مثلا ایا فلان دکمه کارایی درستی دارد ؟ فیلد ها به خوبی پر می‌شوند اررورها به درستی نمایش داده می‌شوند و کارهایی از این جنس

د) همانطور که از اسم این روش پیداست توسعه آزمون محور روشی از توسعه است که ما توسعه را برای پاس شدن تست ها انجام می دهیم.

شما فرض کنید که یک سلسله از نیازمندی ها در برنامه خود دارید حالا میتوانید با نوشتن تست های مختلف به نحوی که تمام نیازمندی های شما را دربرگیرد پا در مسیر tdd بگذارید. سپس توسعه نرم افزار خود را شروع میکنید . تست ها را اجرا می کنید و می بینید که ایا تست های جدید مربوط به این ویژگی پاس می شوند یا نه. اگر پاس نشدند دوباره به توسعه می پردازیم و اگر شدند کد را ریفکتور میکنیم تا با کدهای کثیفی و شلوغ غیرقابل نگهداری روبرو نشویم.

به صورت خلاصه این روش شامل گام‌های زیر است:

۱- تست جدید را برمبنای ویژگی جدید بنویس

۲- تست را اجرا کن و مطمئن شو که کد قبلی این تست را پاس نمیکند(اگه تست پاس بشه یا ویژگی جدید نیست یا تست رو خوب ننوشتیم)

۳- نرم افزار رو به نحوی توسعه بده که تست ها رو پاس کنه

۴- تست ها رو اجرا کن اگه پاس نشدن به گام سه برو

۵- کد رو ریفکتور کن که بعد از یک مدت کد کثیف غیر قابل نگهداری نداشته باشیم

مزیت اصلی این روش جلوگیری از اضافه کاری در توسعه نرم افزار است همچنین این روش به ما کمک می کند در ابتدای کار نیازمندی ها رو به خوبی بررسی کنیم و از دوباره کاری جلوگیری میکند. همچنان چون بر مبنای هر ویژگی و هر ماژول تست می نویسیم و توسعه می دهیم نرم افزاری با modularity بالا خواهیم داشت. همچنین در یکی از منابع اشاره شده بود که این فرآیند منجر به این می‌شود که مجبور به پیاده سازی solid در کدها شویم که با کمی تفکر به این نتیجه رسیدم که تقریبا مجبور به رعایت هر پنج اصل هستیم.

6.

1.

$(a \wedge b)$

$(\neg a \wedge (c \vee d))$

2.

هر Clause به صورت زیر عبارت P را تعیین می کند:

- Clause اول: اگر a و b هر دو درست باشند، عبارت P درست است.
- Clause دوم: اگر a نادرست باشد و حداقل یکی از c و d درست باشند، عبارت P درست است.

3.

a	b	c	d	$a \wedge b$	$c \vee d$	$\neg a \wedge (c \vee d)$	p
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	1
0	1	0	0	0	0	0	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	1	1

0	1	1	1	0	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	1	0	0	1
1	1	0	1	1	1	0	1
1	1	1	0	1	1	0	1
1	1	1	1	1	1	0	1

.4

Coverage Clause:

- تمامی Clause ها باید حداقل یکبار در تست ها شامل شوند.

- a = 1, b = 1, c = 0, d = 0 $T \vee (F \wedge F) = T$
- a = 0, b = 1, c = 0, d = 1 $F \vee (T \wedge T) = T$
- a = 0, b = 0, c = 1, d = 0 $F \vee (T \wedge T) = T$
- a = 0, b = 0, c = 0, d = 1 $F \vee (T \wedge T) = T$

Coverage Predicate:

- تمامی مقادیر مختلف باید در تست ها شامل شوند.

- a = 0, b = 0, c = 1, d = 0 $F \vee (T \wedge T) = T$
- a = 0, b = 0, c = 0, d = 0 $F \vee (T \wedge F) = F$
- a = 1, b = 0, c = 1, d = 1 $F \vee (F \wedge T) = F$
- a = 1, b = 0, c = 0, d = 0 $F \vee (F \wedge F) = F$
- a = 1, b = 1, c = 1, d = 0 $T \vee (F \wedge T) = T$
- a = 1, b = 1, c = 0, d = 0 $T \vee (F \wedge F) = T$

- a = 0, b = 1, c = 1, d = 0 $F \vee (T \wedge T) = T$
- a = 0, b = 1, c = 0, d = 1 $F \vee (T \wedge T) = T$
- a = 1, b = 0, c = 0, d = 0 $F \vee (F \wedge F) = F$

.7

1- متغیرهای method:

- string: یک لیست از رشته‌ها
- element: یک رشته

2- چهار نمونه از خصوصیتهای ممکن:

- خصوصیت مبتنی بر رابط: اگر دوباره همان رشته در لیست وجود داشته باشد، مقدار true باید برگردانده شود. (Repetitive Elements)
- خصوصیت مبتنی بر رابط: اگر عنصری در لیست وجود نداشته باشد، مقدار false باید برگردانده شود. (Non-Existent Element)
- خصوصیت مبتنی بر عملکرد: اگر لیست تهی باشد، مقدار false باید برگردانده شود. (Empty List)
- خصوصیت مبتنی بر عملکرد: اگر عنصر ورودی یک رشته خالی باشد، مقدار false باید برگردانده شود. (Empty String)

3- خصوصیات بلاک بندی شده:

- تهی بودن list:
  - list = null
  - expected output: throw NullPointerException
- تهی بودن عنصر ورودی:
  - [""] = string
  - "" = element
  - expected output: false
- وجود عنصر در لیست:
  - ["string = ["foo", "bar", "baz
  - "element = "bar

- expected output: true
- عدم وجود عنصر در لیست:
  - ["string = ["foo", "bar", "baz
  - "element = "qux
  - expected output: false

4- نیازمندی‌های آزمون برای پوشش Choice Basic:

- تست مواردی که عنصر ورودی در لیست وجود دارد و از نوع مختلفی از رشته‌ها هستند، مانند رشته‌های خالی، رشته‌های با تعداد کاراکترهای مختلف، رشته‌های با کاراکترهای بزرگ و کوچک و ...
- تست مواردی که عنصر ورودی در لیست وجود ندارد و از نوع مختلفی از رشته‌ها هستند، مانند رشته‌های خالی، رشته‌های با تعداد کاراکترهای مختلف، رشته‌های با کاراکترهای بزرگ و کوچک و ...
- تست مواردی که لیست تهی است.
- تست مواردی که عنصر ورودی یک رشته خالی است.

8.

1.

```
IF (a >= b AND a <= c) OR (a >= c AND a <= b) THEN
        PRINT a
  ELSE IF (b >= a AND b <= c) OR (b >= c AND b <= a) THEN
        PRINT b
  ELSE
        PRINT c
```

2.

```
IF a > b THEN
        PRINT a - b
  ELSE
```

```
PRINT b - a
```