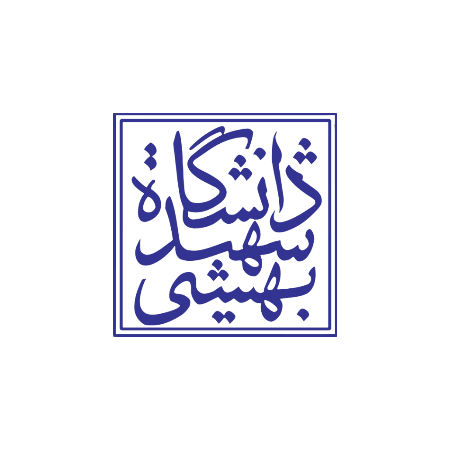
**گرازش تمرین 8**

**بسم الله الرحمن الرحيم**

**درس مبانی سیستم نهفته و بیدرنگ**

**جواب مسئله 1)**

**محاسبه Cyclomatic complexity**

**CC = E – N + 2P**

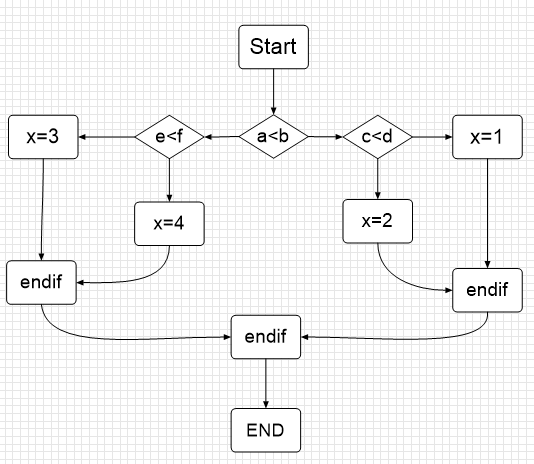
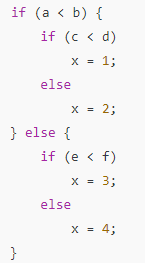
E = # Edges

N = # Nodes

P = # functions (components)

**Simplified formula CC = # Decisions (if statements) + 1**

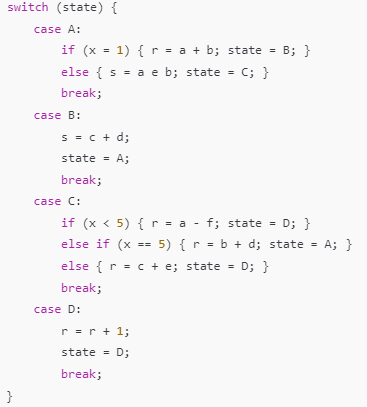
**a)**

****

E = 14 , N = 12, P = 1, # Decisions(if) = 3

CC = 14 – 12 + 2 =4 CC = 3 + 1 = 4

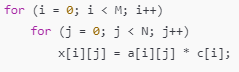
**b)**



# Decisions = 1 switch(state) + 1 if in case A + 2 if in case C = 4

CC = 4 + 1 = 5

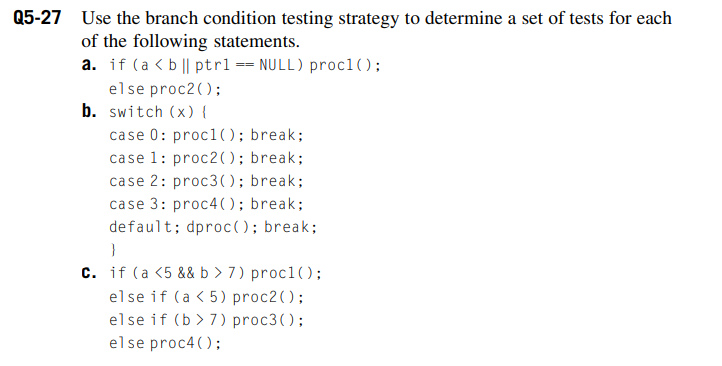
**C)**



# Decisions = 1 in outer for + 1 in inner for = 2

CC = 2 + 1 = 3

**مسئله2)** 5-27 مرجع Wolf را در رابطه با آزمون شرط های انشعاب برای کدهای داده شده حل کنید) بخش 5-10 را پیش از حل مسئله مطالعه کنید (.



**جواب مسئله 2)**

برای حل کردن این مسئله اول بهتر است در مورد آزمون شرط های انشعاب اشاره داشته باشیم.

آزمون وضعیت انشعاب (Branch Condition Testing) یکی از تکنیک‌های آزمون مبتنی بر کنترل جریان (Control Flow Testing) در مهندسی نرم‌افزار است. این آزمون برای بررسی دقیق تمامی مسیرهای ممکن در کد استفاده می‌شود و هدف آن پوشش دادن تمامی وضعیت‌ها و نتایج منطقی مرتبط با شرایط انشعاب (مانند if, switch-case و غیره) است.

#### بخش آ)

   if (a < b || ptr1 == NULL)

        proc1();

    else

        proc2();

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case | a | b | ptr1 | Condition 1 (a < b) | Condition 2 (ptr1 == NULL) | Logical Expression | Action |
| TC 1 | **3** | **5** | **NULL** | **TRUE** | **TRUE** | **TRUE** | **proc1()** |
| TC 2 | **3** | **5** | **Not NULL** | **TRUE** | **FALSE** | **TRUE** | **proc1()** |
| TC 3 | **7** | **5** | **NULL** | **FALSE** | **TRUE** | **TRUE** | **proc1()** |
| TC 4 | **7** | **5** | **Not NULL** | **FALSE** | **FALSE** | **FALSE** | **proc2()** |

#### بخش ب)

|  |  |  |
| --- | --- | --- |
| Test Case | x | Action |
| Test Case 1 | **0** | **proc1()** |
| Test Case 2 | **1** | **proc2()** |
| Test Case 3 | **2** | **proc3()** |
| Test Case 4 | **3** | **proc4()** |
| Test Case 5 | **5** | **dproc()** |

#### 

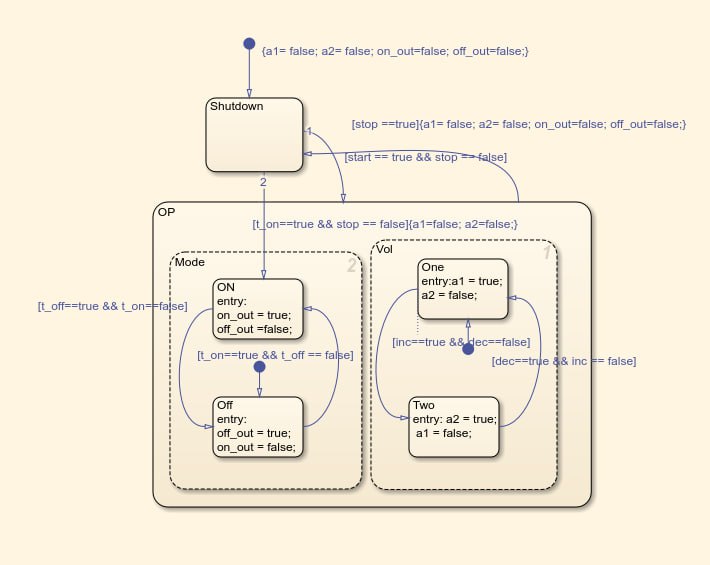
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | a | b | Condition 1 (a < 5) | Condition 2 (b > 7) | Action |
| Test Case 1 | **3** | **8** | **TRUE** | **TRUE** | **proc1()** |
| Test Case 2 | **3** | **6** | **TRUE** | **FALSE** | **proc2()** |
| Test Case 3 | **6** | **8** | **FALSE** | **TRUE** | **proc3()** |
| Test Case 4 | **6** | **6** | **FALSE** | **FALSE** | **proc4()** |

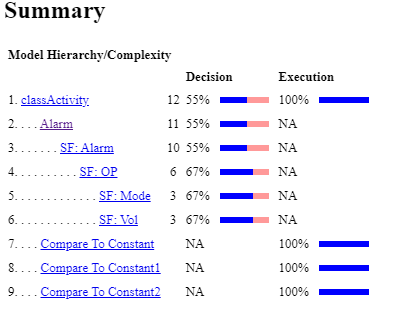
#### بخش ث)

**مسئله3)**

بخش الف) برای این مدل با فرض آزمون با همان ترتیب ورودی مشخص شده در صورت سوال گزارش های مربوط به coverageرا در Stateflow/Simulink استخراج کرده و در گزارش ذکر کنید. چه درصدی از حالتها و گذارها مورد آزمون قرار گرفته است؟

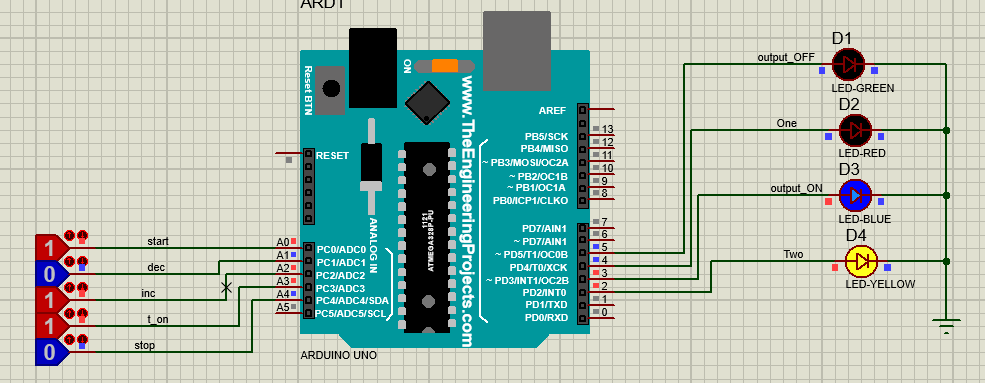
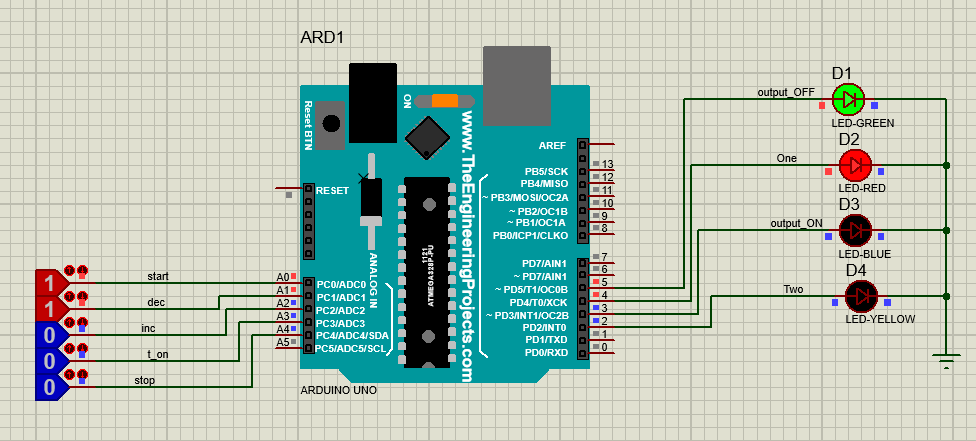
الف) **ماشین حالت تمرین 4**

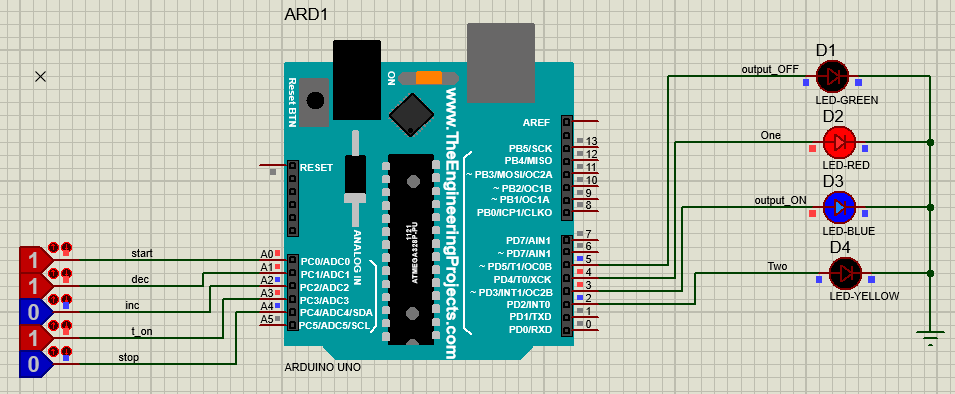
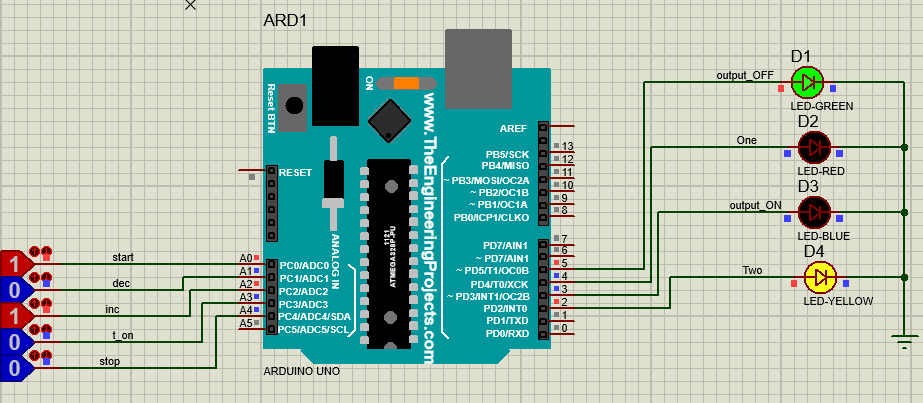


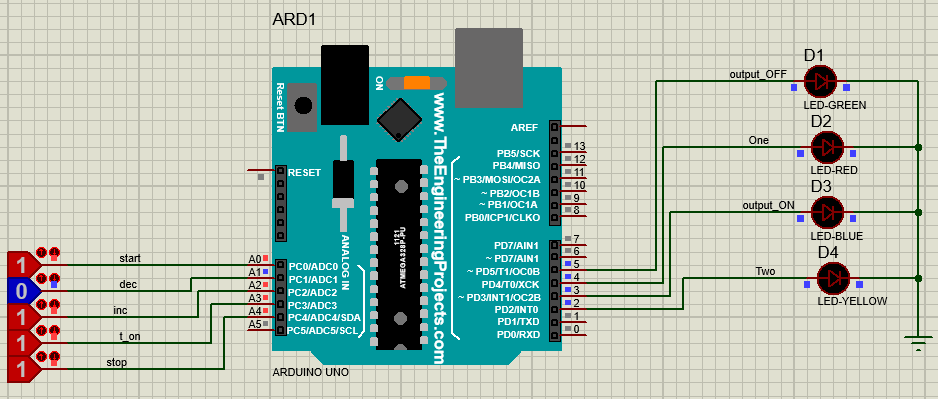


1. **ساختار مدل (Hierarchy)**:
   * مدل به صورت سلسله مراتبی سازمان‌دهی شده است:
     + **classActivity** بخش بالاترین سطح مدل است و شامل زیربخش‌هایی مانند:
       - **Alarm** زیرمجموعه آن
       - **OP** حالات عملیاتی
       - **Mode** حالت‌ها
       - **Vol** حالات ولوم/حجم
2. **متریک‌های پوشش**:
   * **پوشش تصمیم‌گیری (Decision Coverage)**:
     + این متریک درصد تصمیم‌گیری‌های (شاخه‌های منطقی) تست‌شده در طول شبیه‌سازی را نشان می‌دهد.
     + به عنوان مثال، **classActivity** دارای**55% پوشش تصمیم‌گیری** است، یعنی فقط 55% از نقاط تصمیم‌گیری در تست مورد بررسی قرار گرفته‌اند.
   * **پوشش اجرا (Execution Coverage)**:
     + این متریک درصد عناصر مدل (مثل بلوک‌ها، حالات یا انتقال‌ها) که در طول شبیه‌سازی اجرا شده‌اند را نشان می‌دهد.
     + به عنوان مثال، پوشش اجرا برای **classActivity**  برابر با **100%** است، یعنی تمام عناصر این بخش اجرا شده‌اند.
3. **جزئیات بخش‌ها**:
   * **SF: OP**، **SF: Mode** و **SF: Vol** هرکدام دارای **67% پوشش تصمیم‌گیری** هستند، که نشان می‌دهد این بخش‌ها بیشتر از classActivity و Alarm تست شده‌اند.
   * بلوک‌هایی مثل Compare to Constant دارای **100% پوشش اجرا** هستند، به این معنا که این بلوک‌ها به طور کامل تست شده‌اند، هرچند که پوشش تصمیم‌گیری برای آن‌ها قابل اعمال نیست.

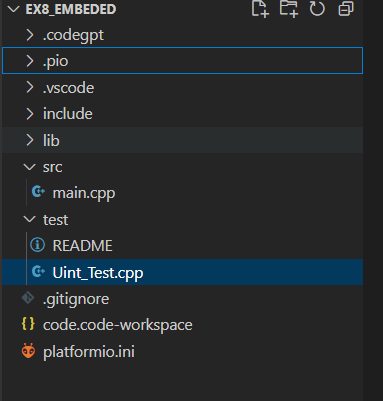
**ب)** شبیه ساز پروتئوس

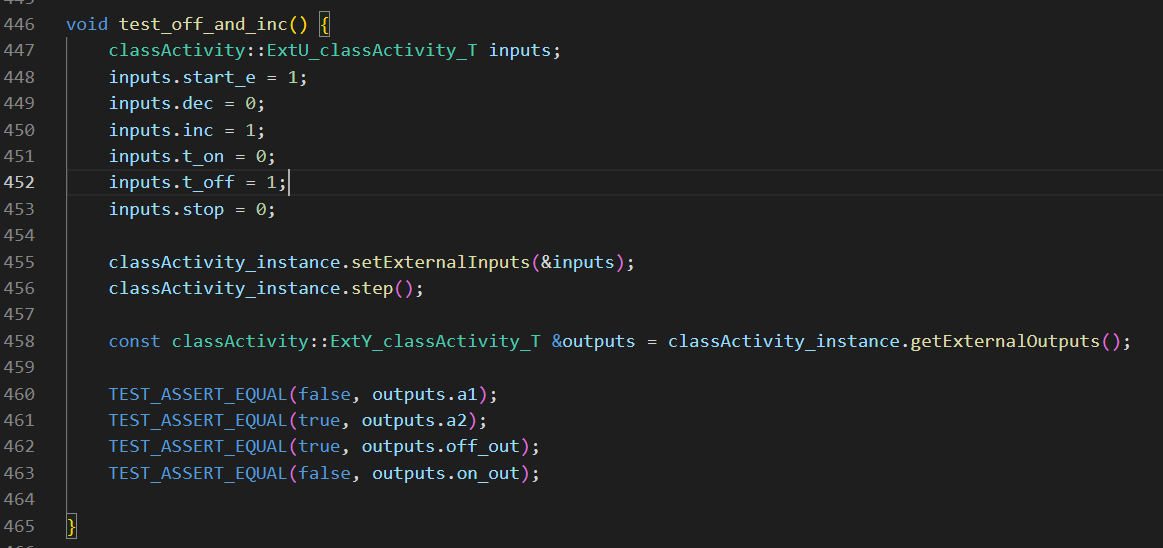




**Test unit)**

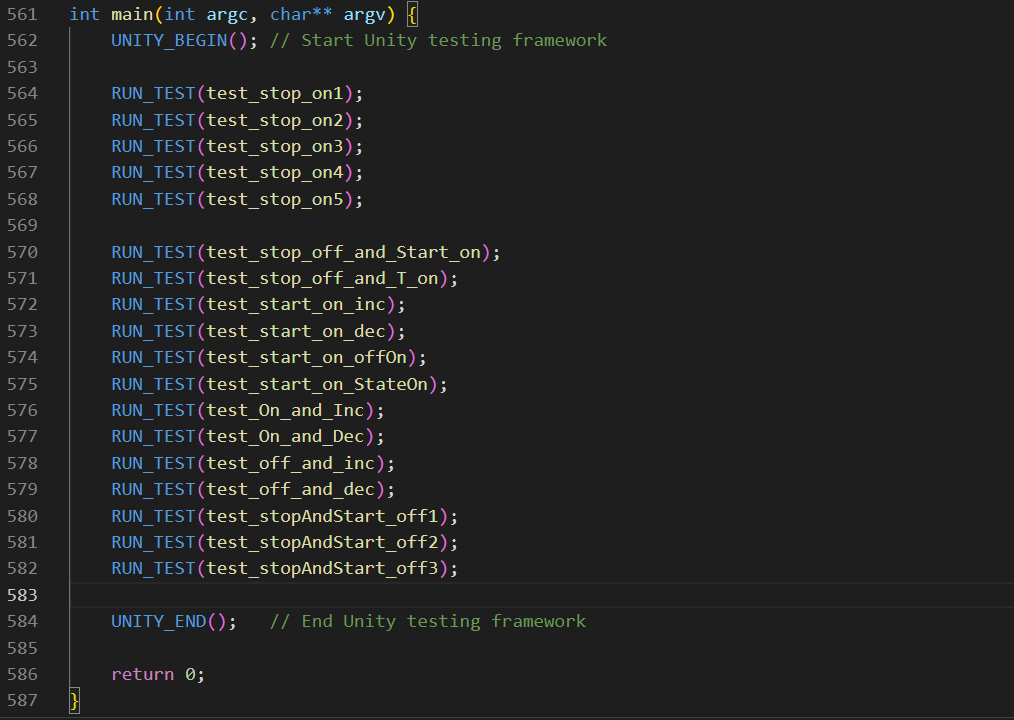


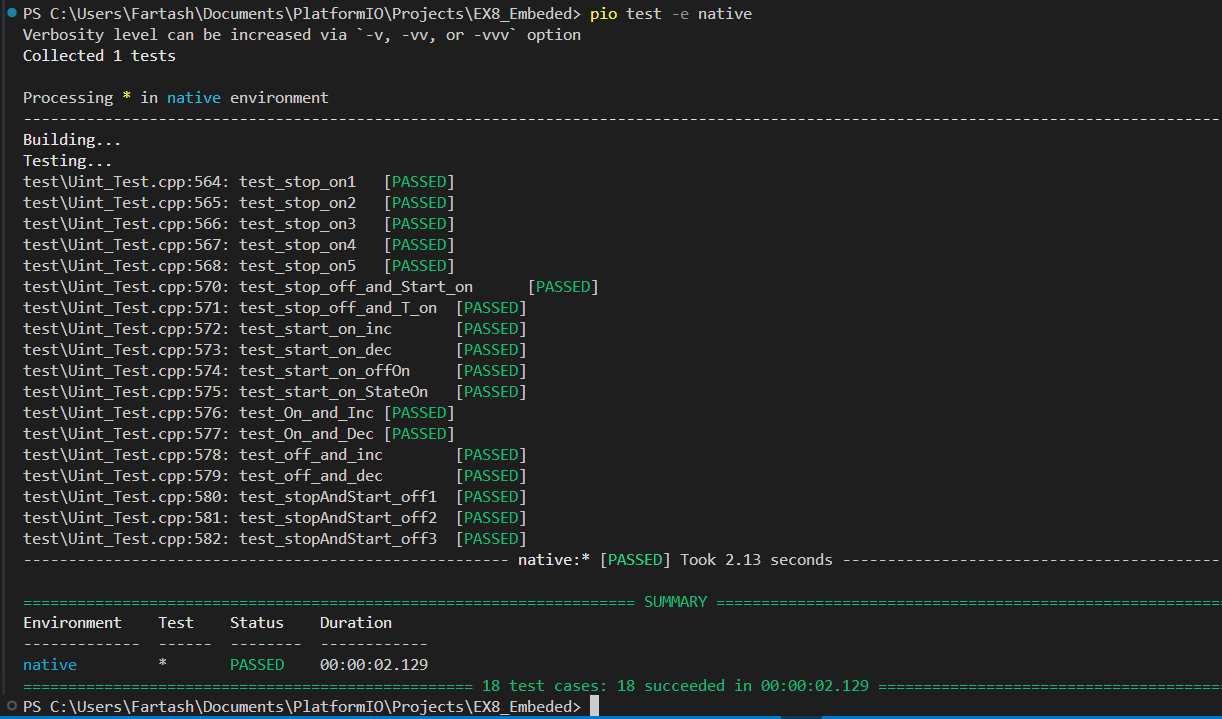
در این فایل همه حالت ها تست شده است که بطور نمونه تست حالت خاموش کردن و اضافه کردن بصورت زیر است:



در این تست, اول ورودی ها را بصورت دستی مقدار دهی می کنیم بعد از فراخوانی تابع step() خروجی ها را تست میکنیم این تابع را در تابع اصلی با استفاده از RUN\_TEST(…) اجرا میکنیم.

تابع اصلی:



**خروجی تست ها:**  


**ج)** معیار ضریب اسپاگتی ((SF ) Spaghetti Factor )معرفی شده در اسلایدهای درس را برای کد تولید شده توسط ابزار به صورت دستی محاسبه کنید

Spaghetti Factor(SF) = SCC + (Globals\*5) + (SLOC/20)

• SCC = Strict Cyclomatic Complexity

• Globals = # of read/write global variables referenced

• SLOC = # source lines of code (e.g., C statements)

**SCC**

* SCC = E – N + 2P
* SCC = #Decisions + 1 #Decisions in code(if, if else, branch , loop) = 11
* SCC = 11 + 1 = 12

**Globals**

* classActivity\_DW (read/write).
* classActivity\_U (read).
* classActivity\_Y (write).

Total Globals = 3.

**SLOC**

Source Lines of Code (SLOC) are the number of executable lines. We exclude comments, declarations, and blank lines:

* From the provided code, count the actual C statements (e.g., assignments, conditionals).
* Approximation: 100 lines of executable code( SLOC 100)

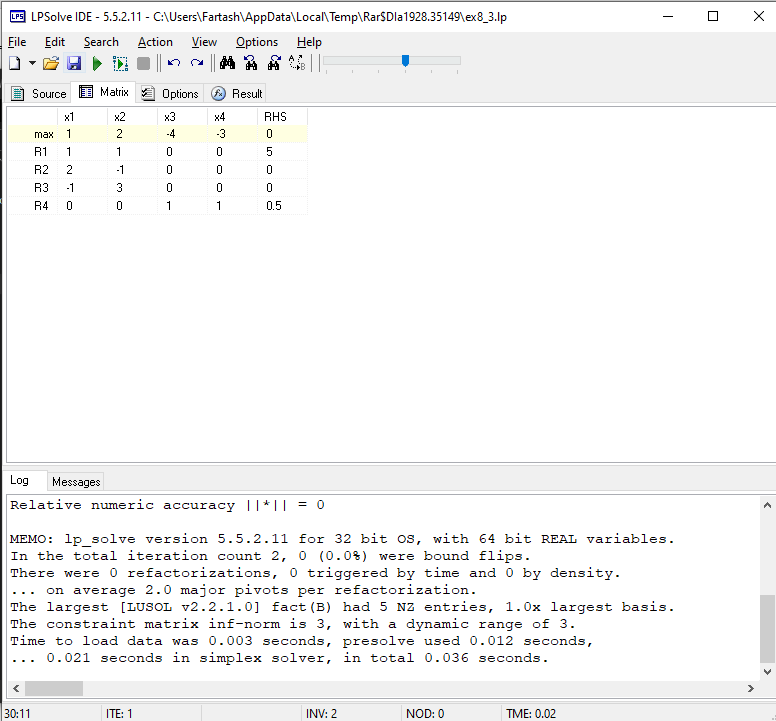
**Spaghetti Factor(SF) = 12 + 5\* 3 + (100/20) = 32**

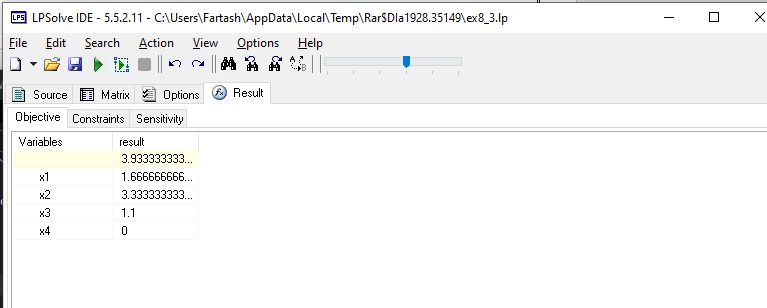
با SF = 32، کد پیچیدگی بالایی را نشان می دهد و می تواند از ساده سازی و ساختار بهتر برای بهبود قابلیت نگهداری بهره مند شود.

#### 

#### بخش امتیازی قسمت الف)

#### 





بخش دوم:

