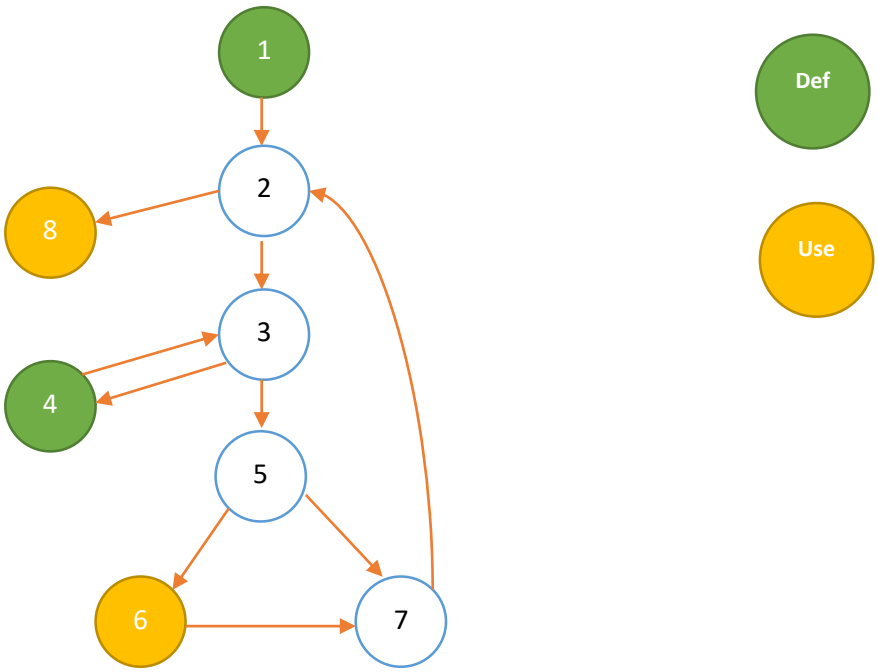
تمرین شماره ۲ – درس آزمون نرم‌افزار

دانشجویان: رامتین باقری، ش.د: ۹۹۳۰۱۹۳۸ – سید سجاد میرزابابایی، ش.د: ۹۹۲۱۰۱۴۲ – امیرحسین کارگران خوزانی، ش.د: ۹۹۲۰۱۱۱۹

سوال اول:

آ) گراف مشخص شده به شکل زیر است:



ب) du-path ها به صورت زیر است:

1. Def(1)Use(8) : [1,2,8]
2. Def(4)Use(6) : [1,2,3,5,6]
3. Def(4)Use(6) : [4,3,5,6]
4. Def(4)Use(8) : [4,3,5,7,2,8]
5. Def(4)Use(8) : [4,3,5,6,7,2,8]

ج) مسیرهای آزمون به صورت r_1-r_6 در ذیل نشان داده شده است. برای هر یک از این مسیرها شماره du-path بر اساس آنچه در قسمت ب) شماره گذاری شده است نمایش داده شده است.

- r_1 : 1
- r_2 : 1
- r_3 : 1 - 2
- r_4 : 1 - 4
- r_5 : 1 - 2 - 3 - 4 - 5
- r_6 : 1 - 2 - 3 - 4 - 5

د) در این پوشش تنها باید مطمئن شویم هر definition حداقل به یک use می‌رسد؛ بنابرین مسیر آزمون r_4 به تنهایی این شرط پوشش را ارضا می‌کند. که در آن definition های 1 و 4 را به use شماره 8 می‌رساند.

r_4 = [1,2,3,4,3,5,7,2,8]

ه) پوشش all-uses سخت گیرانه‌تر عمل می‌کند و می‌گوید هر definition باید به تمامی use های خود برسد که مسیر آزمون r_5 و r_6 این پوشش را ارضا می‌کنند؛ هرچند r_5 حداقل مجموعه آزمون با کوتاه ترین اندازه از میان مسیر های داده شده است.

r_5 = [1,2,3,4,3,4,3,5,6,7,2,8]

و) این پوشش سخت گیرانه ترین حالت ممکن است و می گوید باید تمام مسیر های میان definition ها و use ها پوشش داده شوند و از میان path های داده شده r_6 تمامی مسیر ها را پوشش می‌دهد.

r_6 = [1,2,3,4,3,5,7,2,3,5,6,7,2,8]

سوال دوم:

آ) کلاز[1] یک مسند بدون عملگرهای منطقی است، بنابراین a, b, c تنها کلازهای مسند P هستند.

ب) هر کدام از کلاز می‌تواند دو مقدار True یا False را اختیار کند، بنابراین به ازای True یا False بودن هر کدام، مسند P طبق جدول زیر تغییر پیدا می‌کند:

	True	False
a	$P: b \vee c$	$P: b \wedge c$
b	$P: a \vee c$	$P: a \vee c$
c	$P: a \vee b$	$P: a \vee b$

همچنین با اصلی[2] در نظر گرفتن هر یک از مسندها، مسندهای نظیر هر کلاز به صورت زیر خواهد بود:

$$Pa: (\sim b \wedge c) \vee (\sim c \wedge b)$$

$$Pb: (\sim a \wedge c) \vee (\sim c \wedge a)$$

$$Pc: (\sim a \wedge b) \vee (\sim b \wedge a)$$

برای بدست آوردن مسندهای نظیر هر کلاز کافیست، کلازهای دیگر به صورتی در نظر گرفته شوند که کلازی که major در نظر گرفته شده تنها تعیین کننده مقدار مسند P باشد.

ج) در قسمت قبل مسندهای نظیر هر کلاز بدست آمد، جدول درستی به صورت زیر خواهد بود:

شماره سطر	a	b	c	P	Pa	Pb	Pc
1	T	T	T	T			
2	T	T		T	T	T	
3	T		T	T	T		T
4	T					T	T
5		T	T	T		T	T
6		T			T		T
7			T		T	T	
8							

[1] Clause
[2] Major

د) زوج سطرهای زیر، پوشش GACC را بر اساس جدول درستی ارضا می‌کند:

کلاز اصلی	مجموعه آزمون‌های ممکن
a	(2,6), (2,7), (3,6), (3,7)
b	(2,4), (2,7), (5,4), (5,7)
c	(3,4), (3,6), (5,4), (5,6)

ه) زوج سطرهای زیر، پوشش RACC را بر اساس جدول درستی ارضا می‌کند:

کلاز اصلی	مجموعه آزمون‌های ممکن
a	(2,6), (3,7)
b	(2,4), (5,7)
c	(3,4), (5,6)

و) تاپل‌های چهارتایی به صورت دو زوج سطر True و False در جدول زیر، پوشش RICC را بر اساس جدول درستی ارضا می‌کند و هیچ کدام infeasible نیست.

کلاز اصلی	مجموعه آزمون‌های ممکن	
a	P = T: (1,5)	P = F: (4,8)
b	P = T: (1,3)	P = F: (6,8)
c	P = T: (1,2)	P = F: (7,8)

ز) تاپل‌های چهارتایی به صورت دو زوج سطر True و False در جدول زیر، پوشش GICC را بر اساس جدول درستی ارضا می‌کند و هیچ کدام infeasible نیست.

کلاز اصلی	مجموعه آزمون‌های ممکن	
a	P = T: (1,5)	P = F: (4,8)
b	P = T: (1,3)	P = F: (6,8)
c	P = T: (1,2)	P = F: (7,8)

سوال سوم:

آ) جدول کارنو برای تابع f به صورت زیر خواهد بود:

$$f = ab + a{\sim}bc + {\sim}a{\sim}bc$$

	$\sim a\sim c$	$a\sim c$	ac	$\sim ac$
$\sim b$	0	0	1	1
b	0	1	1	0

و در نتیجه خروجی آن به صورت زیر خلاصه می‌شود:

$$f = ab + {\sim}bc$$

جدول کارنو برای تابع $\sim f$ به صورت زیر خواهد بود:

	$\sim a\sim c$	$a\sim c$	ac	$\sim ac$
$\sim b$	1	1	0	0
b	1	0	0	1

و در نتیجه خروجی آن به صورت زیر خلاصه می‌شود:

$$\sim f = {\sim}b{\sim}c + {\sim}ab$$

ب) implicant ها در قسمت آ) تعیین شده است، همانگونه که از جدول و عبارت بدست آمده مشخص است، تعداد Non-redundant انها برای هر یک از توابع f , $\sim f$ عدد ۲ می‌باشد.

ج) براساس DNF توابع f , $\sim f$ مجموعه implicant ها به صورت زیر خواهد بود:

$$Implicants: \{ ab, \sim bc, \sim b \sim c, \sim ab \}$$

	a	b	c
ab	T	T	
$\sim bc$		F	T
$\sim b \sim c$		F	F
$\sim ab$	F	T	

نیازمندی‌های آزمون شامل تمام implicantهایی است که مقدار آن‌ها برابر True است:

Possible test set for f: {TTT, TTF, FTF, FFT}

در ادامه برای پاسخ دهی به قسمت‌های ج، د، و، ه از مقدار 1 برای True و از 0 برای False استفاده شده است.

د) MUTP:

Test Data:

110 - UTP for term ab

111 - UTP for term ab

001 - UTP for term !bc

101 - UTP for term !bc

MUTP is feasible for all terms.

Number of tests: 4

Number and types of mutants generated:

Number of False mutants generated: 0

Number of True mutants generated: 0

Number of TOF mutants generated: 0

Number of TIF mutants generated: 0

Number of TRF mutants generated: 4

Total Number of Non-Equivalent Mutants Generated: 4

Mutants:

a & b & c | !b & c

a & b & !c | !b & c

a & b | !b & c & a

a & b | !b & c & !a

:CUTPNFP (٥

Test Data:

110 - UTP for term ab

010 - NFP for literal a in term ab

100 - NFP for literal b in term ab

001 - UTP for term !bc

011 - NFP for literal b in term !bc

000 - NFP for literal c in term !bc

CUTPNFP is feasible for all literals.

Number of tests: 6

Number and types of mutants generated:

Number of False mutants generated: 0

Number of True mutants generated: 0

Number of TOF mutants generated: 0

Number of TIF mutants generated: 4

Number of TRF mutants generated: 2

Total Number of Non-Equivalent Mutants Generated: 6

Mutants:

a & b & c | !b & c

a & b | !b & c | !a & b & !c

a & b | !b & c | a & !b & !c

a & b | !b & c & a

a & b | !b & c | !a & b & c

a & b | !b & c | !a & !b & !c

و) MNFP:

Test Data:

010 - NFP for literal a in term ab

011 - NFP for literal a in term ab, literal b in term !bc

100 - NFP for literal b in term ab, literal c in term !bc

000 - NFP for literal c in term !bc

MNFP is not feasible for all literals.

Number of tests: 4

Number and types of mutants generated:

Number of False mutants generated: 0

Number of True mutants generated: 0

Number of TOF mutants generated: 0

Number of TIF mutants generated: 4

Number of TRF mutants generated: 0

Total Number of Non-Equivalent Mutants Generated: 4

Mutants:

a & b | !b & c | !a & b & !c

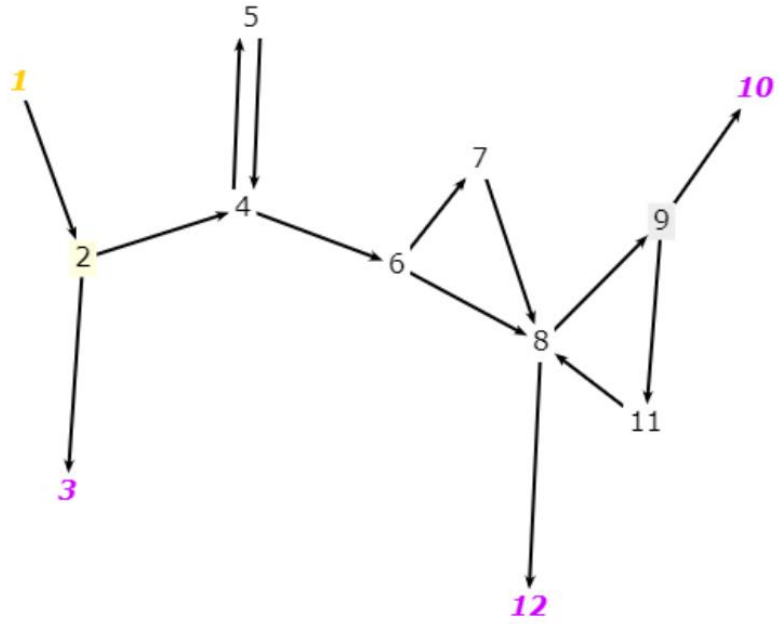
a & b | !b & c | !a & b & c

a & b | !b & c | a & !b & !c

a & b | !b & c | !a & !b & !c

سوال چهارم:

رسم گراف تابع myAtoi به وسیله مشخص کردن قسمت‌های برنامه (عبارت‌های ساده، عبارت‌های شرطی، حلقه‌ها و ...) به عنوان نودهای گراف و سپس تعیین جریان‌های ممکن در تابع به وسیله یال‌های میان این نودها است. نتیجه این فرآیند، گراف تابع مذکور خواهد بود که در شکل زیر قابل رویت است (گره نارنجی رنگ: گره شروع):



تصویر 1 - گراف تابع myAtoi (نارنجی: آغازین، بنفش: پایانی)

پس از تشکیل گراف، می‌توانیم مسیرهای پوشش ساده و پرایم را از آن استخراج کنیم. همانطور که از تعریف برمی‌آید، مسیرهای پرایم، مسیرهای ساده‌ای هستند که زیرمسیر هیچ مسیر ساده دیگری نیستند. در گراف مورد نظر، ۹۰ مسیر ساده وجود دارند که از میان آنها ۲۱ مسیر از نوع پرایم هستند.

۱	1,2,4,6,7,8,9,10	۲	1,2,4,6,7,8,9,11	۳	1,2,4,6,8,9,10	٤	1,2,4,6,8,9,11
٥	1,2,4,6,7,8,12	٦	5,4,6,7,8,9,11	۷	5,4,6,7,8,9,10	۸	1,2,4,6,8,12
۹	5,4,6,8,9,11	۱۰	5,4,6,8,9,10	۱۱	5,4,6,7,8,12	۱۲	5,4,6,8,12
۱۳	1,2,4,5	۱٤	9,11,8,12	۱٥	9,11,8,9	۱٦	11,8,9,10
۱۷	11,8,9,11	۱۸	8,9,11,8	۱۹	5,4,5	۲۰	4,5,4
۲۱	1,2,3						

سوال پنجم:

با داشتن مسیرهای پوشش پرایم که در قسمت قبلی بدست آمد، می‌توانیم مسیرهای آزمون را طراحی کنیم. مسیرهای آزمون می‌بایست از یک نود آغازین شروع و به یک نود پایانی ختم شوند (در صورتی که مسیرهای پوشش پرایم لزوما این شرط را ندارند). لذا می بایست مجموعه مسیر های آزمونی ایجاد کنیم که هر مسیر آزمون می‌بایست یک یا چند مسیر پرایم را به عنوان زیر مسیر در خود داشته باشد. هر چه تعداد مسیرهای آزمون ایجاد شده (و به تبع آن تعداد موارد آزمون تولید شده) کمتر باشد (با شرط پوشش کمتر یا مساوی با پوشش ایجاد شده توسط مسیرهای پرایم)، کارایی آزمون بیشتر خواهد شد.

پس از تحلیل مسیرهای پرایم و تلاش برای تبدیل آن‌ها به مسیرهای آزمون، مسیرهای یکتا طبق جدول زیر

شماره	مسیر آزمون	شماره مسیرهای پرایم پوشش داده‌شده
۱	1,2,3	۲۱
۲	1,2,4,5,4,6,8,9,11,8,12	۴، ۹، ۱۳، ۱۴، ۱۸، ۲۰
۳	1,2,4,5,4,6,7,8,12	۵، ۱۱، ۱۳، ۲۰
٤	1,2,4,5,4,6,8,12	۸، ۱۲، ۱۳، ۲۰
٥	1,2,4,5,4,6,7,8,9,11,8,9,11,8,9,10	۲، ۳، ۶، ۷، ۱۰، ۱۳، ۱۵، ۱۶، ۱۷، ۱۸، ۱۹، ۲۰

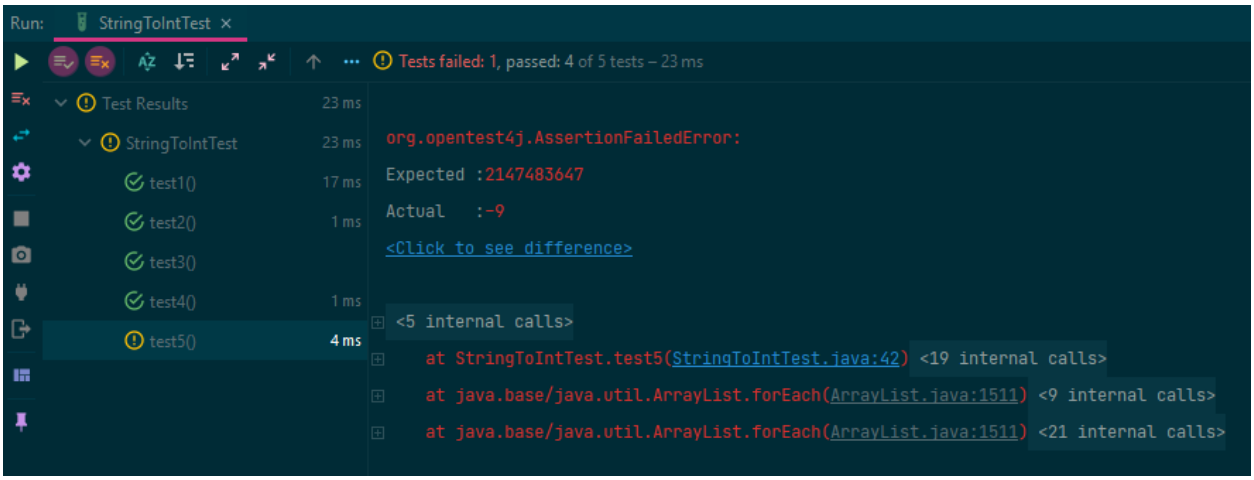
بدست می‌آیند که شماره مسیرهای قرمز رنگ به صورت sidetrip در مسیر آزمون tour شده‌اند. حال می‌بایست بر مبنای این مسیرهای آزمون، موارد آزمون را طراحی کنیم:

شماره مسیر آزمون	مورد آزمون	
	ورودی	خروجی مورد انتظار
۱	“”	0
۲	“ 1”	1
۳	“ -t”	0
٤	“ t”	0
٥	Infeasible	

بعد از بررسی مسیر آزمون شماره ۵ روی کد برنامه، متوجه شدیم که طی مسیر مورد نظر امکان‌پذیر نیست (زیرا برای رفتن به گره شماره ۱۰، می‌بایست متغیر میانی result برابر با $2^{31} - 1$ باشد که این یعنی حلقه ۸-۹-۱۱ می‌بایست به تعداد رقم‌های این عدد طی شود. در اینجا می‌توانیم مسیر شماره ۵ را با نیازمندی موجود تغییر دهیم و جدول بالا را بازنویسی کنیم:

شماره مسیر آزمون	مورد آزمون	
	ورودی	خروجی مورد انتظار
۱	“”	0
۲	“ 1”	1
۳	“ -t”	0
٤	“ t”	0
٥	“ +21474836471”	2147483647

بعد از تهیه موارد آزمون، کد مربوط به اجرای آزمون‌ها را در فایل StringToIntTest نوشتیم. تصویر زیر، نتیجه اجرای آزمون است:



پس از بررسی مسیر آزمون ۵ متوجه می‌شویم که در یکی از خط‌های برنامه (که در مسیر مورد نظر اجرا می‌شود)، شرط نادرستی مورد ارزیابی قرار می‌گیرد. تصویر حاوی این قطعه کد در زیر آمده است:

```
    // 8
    while (i < str.length() && str.charAt(i) >= '0' && str.charAt(i) <= '9') {
        // 9
        if (result > Integer.MAX_VALUE ||
                (result == Integer.MAX_VALUE  && str.charAt(i) - '0' > Integer.MAX_VALUE % 10)) {
            // 10
            return (sign == 1) ? Integer.MAX_VALUE : Integer.MIN_VALUE;
        }
        // 11
        result = result * 10 + (str.charAt(i++) - '0');
    }
    // 12
    return result * sign;
}
```

با توجه به توضیحات موجود در سوال، عبارت شرطی که با خط سبز رنگ مشخص شده است کاملا اشتباه است، زیرا تنها در صورتی موجب خاتمه اجرای برنامه می‌گردد که مقدار متغیر میانی result برابر با Integer.MAX_VALUE و رقم i بزرگتر از رقم آخر Integer.MAX_VALUE (یعنی ۷) باشد. با حذف شرط فوق، می‌توان مشکل این برنامه را رفع کرد.