سوال اول : برنامه findVal و موتانت آن که مشخص شده است را در نظر بگیرید و به سوالات الف تا د پاسخ دهید:

```
// Effects : If numbers null throw NullPointerException
// else return FIRST occurrence of val in numbers[]
// if val not in numbers[] return -1
1. public static int findVal(int numbers[], int val)
2. {
3.       int findVal = -1;
4.
5.       for (int i=0; i < numbers.length; i++)
6.           if (numbers [i] == val)
7.              {   findVal = i;
7.              {   findVal = val;
8.                  return (findval)
9.              }
10.     return (findVal);
11.}
```

الف) در صورت امکان ورودی‌ای بیابید که منجر به عدم Reachability به موتانت شود.

ب) در صورت امکان ورودی‌ای بیابید که منجر به Reachability شود اما Infection روی موتانت رخ ندهد.

ج) در صورت امکان ورودی‌ای بیابید که منجر به Infection شود اما Propagation روی موتانت رخ ندهد.

د) در صورت امکان ورودی‌ای بیابید که منجر به kill موتانت شود.

سوال دوم : متد ()power را در نظر بگیرید:

```
1. Public static int power (int left, int right)
2. {
3. //*********************************************
4. // Raises left to the power of right
5. // precondition : right >= 0
6. // postcondition : Return left**right
7. //*********************************************
8.
9. int rslt;
10. rslt = left;
11. if (right == 0)
12.  {
13.   rslt = 1;
14.  }
15. else
16.  {
17.    for (int i = 2; i <= right; i++)
18.      rslt = rslt * left;
19.  }
20. return (rslt);
21.}
```

الف) با استفاده از عملگرهای موتانت موثر، ۱۲ موتانت برای متد ()power تعریف کنید. (سعی کنید از هر عملگر موتانت حداقل یکبار استفاده نمایید.)

ب) تعداد کل موتانت‌هایی که برای این متد می‌توان ایجاد کرد، تقریباً چقدر است؟

ج) برای موتانت‌های حاصل شده در قسمت الف، موارد آزمونی ایجاد کنید که آن‌ها را بکشد؟ چه تعداد مورد آزمون برای کشتن موتانت‌های غیر معادل لازم است؟

سوال سوم : گرامر ورودی یک برنامه به صورت زیر است:

1. Input ::= statement
2. statement : = expr
3. – 4. expr := LHS RHS AO | AO LHS RHS
5. LHS ::= digit
6. RHS ::= digit
7. – 10. AO ::= “Div” | “Mod” | “GCD” | “LCM”
11. – 20. digit ::= “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”

الف) برای آزمون این برنامه، با پوشش نمادهای پایانه‌ای (TSC) رشته‌های ورودی لازم را بنویسید.

ب) با استفاده از روش موتاسیون یک گرامر جدید برای تولید ورودی‌های نامعتبر بنویسید.

ج) برای آزمون برنامه با استفاده از گرامر جهش یافته قسمت "ب" با پوشش قواعد (PC) رشته‌های ورودی نامعتبر را بنویسید.

سوال چهارم : استاندارد ISO 29119-2 (پارت دوم استاندارد - فرآیند آزمون) را مطالعه و با فرآیند آزمون بیان شده در کتاب مقایسه کنید.

بخش عملی

در این بخش به شما پیاده‌سازی یک کامپایلر برای بخشی از زبان مینی‌جاوا (زیرمجموعه‌ای از زبان جاوا) داده شده است. این زبان شیء؛گراست ولی پشتیبانی از قسمت بسیار محدودی از ویژگی‌های شیءگرایی در این پروژه در نظر گرفته شده است. بنابراین گرامر زبان مینی‌جاوا با ساده‌سازی بسیار مبنای بخش عملی قرار گرفته است. سورس کد به همراه گرامر تغییریافته را از اینجا می‌توانید مشاهده کنید.

پروژه داده‌شده دارای مشخصات زیر است:

- کامپایلر تک‌گذره است و از ۵ بخش تشکیل‌شده است: ۱. تحلیل‌گر لغوی (اسکنر) ۲. تحلیل‌گر نحوی (پارسر) که به روش SLR(1) پیاده‌سازی شده است. ۳. تحلیل‌گر معنایی ۴. مولد کد میانی که کد تولیدشده آن در قالب کدهای ۳ آدرسه (که در انتها آمده است) است. ۵. خطاپرداز
- ورودی کامپایلر یک متن حاوی برنامه‌ای است که کامپایلر شما باید آن را ترجمه کند.

- خروجی کامپایلر شما یک متن حاوی کد میانی تولیدشده است.
- دستور مربوط به چاپ در این زبان، تنها محتوای از جنس عدد صحیح را می‌تواند چاپ کند.
- در نسخه ساده‌شده مینی جاوا، فرض کنید که هیچ اشاره‌ای رو به جلو رخ نخواهد داد (چرا که در این صورت کامپایلر تک‌گذره نخواهد شد).
- اگر یک تابع یا مقدار از رده فرزند مورد استفاده قرار گرفت، تنها در صورتی به رده پدر رجوع شود که رده فرزند آن مقدار را override نکرده باشد.

در این بخش شما به عنوان مهندس تست و کنترل کیفیت نرم‌افزار قرار است از تمام دانشی که در طول ترم یاد گرفتید استفاده کنید و برنامه داده شده را مورد آزمون قرار داده، خطاهای آن را پیدا کنید، و گزارش کار و همچنین کد موارد آزمونی (Junit) که نوشتید را تحویل دهید.

قالب کد میانی:

توضیح	قالب کد سه‌آدرسه
عملوند اول و دوم جمع می‌شوند و در مقصد قرار می‌گیرند.	(ADD, S1, S2, D)
عملوند اول و دوم AND می‌شوند و در مقصد قرار می‌گیرند.	(AND, S1, S2, D)
محتوای مبدأ در مقصد قرار می‌گیرد.	(ASSIGN, S, D)
اگر S1 و S2 مساوی باشند در D مقدار true و در غیر این صورت false ذخیره می‌شود.	(EQ, S1, S2, D)
حاصل S بررسی می‌شود و در صورتی که false باشد به L جهش می‌کند.	(JPF, S, L)
پرش به L انجام می‌شود.	(JP, L)
اگر S1 از S2 کوچکتر باشد مقدار D برابر true و در غیر این صورت مقدار false می‌گیرد.	(LT, S1, S2, D)
عملوند اول در عملوند دوم ضرب می‌شود و در مقصد قرار می‌گیرد.	(MULT, S1, S2, D)
عملوند نقیض می‌شود.	(NOT, S, D)
محتوا بر روی صفحه چاپ می‌شود.	(PRINT, S)
عملوند دوم از عملوند اول کم می‌شود و در مقصد قرار می‌گیرد.	(SUB, S1, S2, D)

*** از مدهای نشان دهی مستقیم (مانند t۱) یا غیر مستقیم (با گذاشتن @ در ابتدای عملوند مانند @t۱) و مقدار صریح (با گذاشتن # در ابتدای عملوند مانند #۵) می توانید در کد میانی استفاده کنید.**

*** میزان فضای در نظر گرفته شده برای متغیرها (مثل t۱) همگی برابر ۴ بایت است.**

*** دستور PRINT تنها محتوای از جنس عدد را چاپ می کند.**