مدرس: دکتر احسان خامس‌پناه
طراحان: پاشا براهیمی، سامان اسلامی نظری، آرش شاهین

بخش تئوری

سوال اول

با توجه به کد داده شده، به سوالات زیر پاسخ دهید.

```
int getAverageOrderQuantityByCustomer(int customer) {
	var sum = 0;
	var count = 0;

	for (Order oldOrder : orderHistory) {
		if (oldOrder.customer == customer) {
		sum += oldOrder.quantity;
		count++;
		}
	}

	if (orderHistory.size() == 0) {
		return 0;
	}

	return sum / count;
}
```

الف) برای قطعه کد بالا CFG را رسم کنید. سپس تمام Prime Path-ها و DU Path-ها را لیست کنید.

ب) مجموعه آزمایه‌ای تولید کنید که تمامی DU Path-ها و Prime Path-ها را پوشش دهد.

ج) آیا کد ما خطایی دارد؟ اگر بله، توضیح دهید در کدام آزمایه این خطا دیده می‌شود و چگونه می‌توان آن را رفع کرد.

پاسخ:

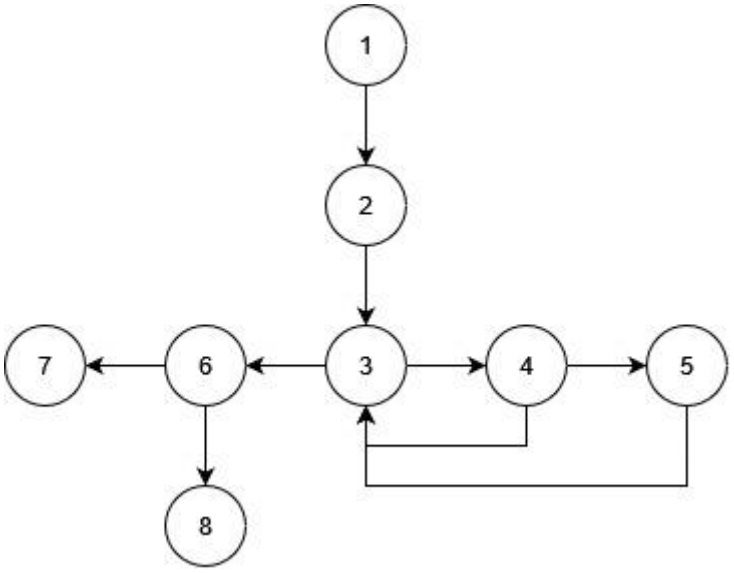
الف)

```
int getAverageOrderQuantityByCustomer(int customer) {
      var sum = 0;
      var count = 0;

      for (Order oldOrder : orderHistory) {
            if (oldOrder.customer == customer) {
                  sum += oldOrder.quantity;
                  count++;
            }
      }

      if (orderHistory.size() == 0) {
            return 0;
      }

      return sum / count;
}
```



(CFG)

Prime Paths: {1, 2, 3, 6, 7} {1, 2, 3, 6, 8} {3, 4, 3} {3, 4, 5, 3} {4, 5, 3, 4} {4, 5, 3, 6, 7} {4, 5, 3, 6, 8} {4, 3, 6, 7} {4, 3, 6, 8} {5, 3, 4, 5}

Infeasible: {4, 3, 6, 7} {1, 2, 3, 6, 8} {4, 5, 3, 6, 7}

Definition Use:

Node	Definition	Use
1	customer, sum, count, orderHistory	
2	oldOrder	
3		
4		
5	sum, count	sum, count, oldOrder
6		
7		
8	sum, count	

Edge	Use
(3,4)	orderHistory, oldOrder
(3,6)	orderHistory, oldOrder
(4,3)	oldOrder, customer
(4,5)	oldOrder, customer
(6,7)	orderHistory
(6,8)	orderHistory

Variable	Definition Use Pairs
customer	(1, (4,3)) (1, (4,5))
sum	(1, 5) (5, 5) (1, 8) (5, 8)
count	(1, 5) (5, 5) (1, 8) (5, 8)
orderHistory	(1, (3,4)) (1, (3,6)) (1, (6,7)) (1, (6,8))
oldOrder	(2,5) (2, (3,4)) (2, (3,6)) (2, (4,3)) (2, (4,5))

Variable	Definition Use Paths
customer	(1, 2, 3, 4, 3) (1, 2, 3, 4, 5)
sum	(1, 2, 3, 4, 5) (5, 3, 4, 5) (1, 2, 3, 6, 8) (5, 3, 6, 8)
count	(1, 2, 3, 4, 5) (5, 3, 4, 5) (1, 2, 3, 6, 8) (5, 3, 6, 8)
orderHistory	(1, 2, 3, 4) (1, 2, 3, 6) (1, 2, 3, 6, 7) (1, 2, 3, 6, 8)
oldOrder	(2 ,3 ,4, 5) (2, 3, 4) (2, 3, 6) (2, 3, 4, 3) (2, 3, 4, 5)

Unique Paths: (1, 2, 3, 4, 3) (1, 2, 3, 4, 5) (5, 3, 4, 5) *(1, 2, 3, 6, 8)* (5, 3, 6, 8) (1, 2, 3, 6) (1, 2, 3, 6, 7) (2 ,3 ,4, 5) (2, 3, 4) (2, 3, 6) (2, 3, 4, 3) (2, 3, 4, 5)

Infeasible: {1, 2, 3, 6, 8}

ب)

1. 12367 (Order history = {}, customer = 1)
2. 12345345368 (Order history = {1, 1}, customer = 1)
3. 1234368 (Order history = {2}, customer = 1)

ج) در آزمایه سوم می‌بینیم که به ارور می‌خوریم زیرا count ما 0 می‌باشد و ارور تقسیم بر 0 رخ می‌دهد. پس با آزمایه‌ها توانستیم یک ارور را رفع کنیم. کافی است که در این قسمت کد:

```
if (orderHistory.size() == 0)
```

یکی از این دو تغییر را ایجاد کنیم:

```
if (count == 0)
if (orderHistory.size() == 0 || count == 0)
```

یا اینکه یک if جدید درست کنیم:

```
if (count == 0) {
    return 0;
}
```

سوال دوم

قطعه کد زیر را در نظر بگیرید:

```
var1, var2 = false, false;

if (a | b) {
  var1 = true;
} else if (c) {
  var2 = true;
```

```
}

return var1, var2; // Returns the pair of (bool, bool)
```

الف) حداقل چند آزمایه برای پوشش جمله صد درصدی و چند آزمایه برای پوشش شاخه صد درصدی این تکه کد نیاز است؟ (صرفا حالات مقداردهی متغیرهای a و b و c را بنویسید)
ب) آیا می‌توانید کد را به صورتی تغییر دهید که خروجی و منطق تابع تغییری نکند ولی بتوان با یک آزمایه به پوشش جمله صد درصدی رسید؟
ج) بر اساس CFG توابع، یک شرط کافی برای اینکه بتوان با یک آزمایه به پوشش جمله صد درصدی رسید ارائه کنید.

پاسخ:

الف) برای پوشش کامل جمله‌ها باید داخل هر دو شرط برویم اما از آنجایی که شرط دوم نیازمند نقیض شرط اول است، باید حداقل دو آزمایه داشته باشیم تا هر دو حالت شرط اول را ببینیم:

```
a = true, b = false or true, c = false or true
a = false, b = false, c = true
```

برای اینکه پوشش کامل شاخه‌ها را داشته باشیم، باید حداقل یک بار شرط شاخه درست و یک بار هم غلط باشد. یک آزمایه دیگر برای اینکه حالت غلط شرط دوم را ببینیم نیاز است:

```
a = false, b = false, c = false
```

ب) تنها در صورتی که شرط $\neg(a \vee b) \wedge c$ برقرار باشد مقدار var2 برابر True خواهد شد. در نتیجه در صورتی که شرط $\neg(\neg(a \vee b) \wedge c) \equiv (a \vee b) \vee \neg c$ برقرار باشد مقدار var2 برابر False خواهد بود. بنابراین نقیضِ نقیض این شرط به صورت زیر خواهد بود:

```
var1, var2 = false, true;

if (a | b) {
  var1 = true;
}

if (a | b | (not c)) {
  var2 = false;
}

return var1, var2;
```

از آنجایی که شروط var1 تغییری نکرده و شروط var2 نقیض نقیض قبلی است، منطق برنامه تفاوتی نکرده. با این حال اکنون می‌توانیم با یک آزمایه به صورت زیر به پوشش کامل جملات برسیم:

```
a = true, b = false or true, c = false or true
```

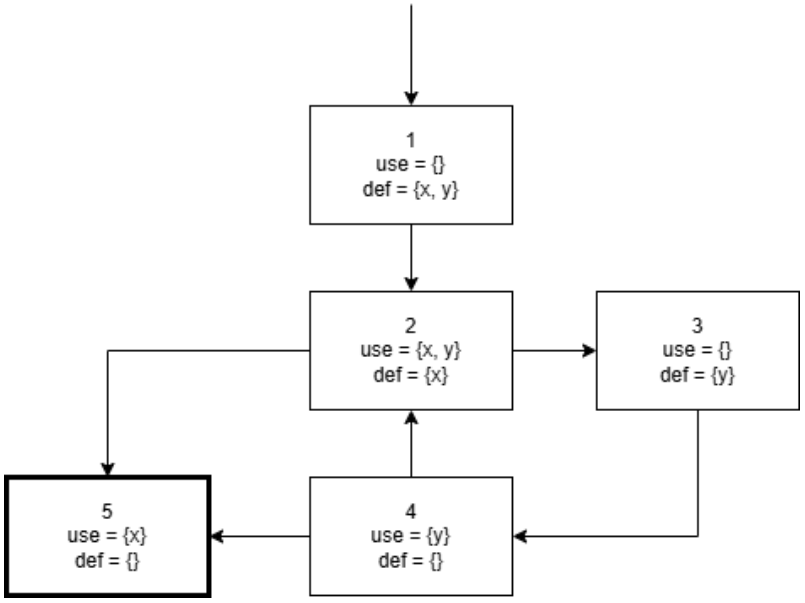
ج) برای اینکه با یک آزمایه به پوشش کامل جملات برسیم، CFG برنامه مورد نظر باید شامل یک مسیری باشد که در آن تمام گره‌ها موجود هستند.

سوال سوم

اثبات کنید که پوشش کامل مسیرهای du لزومی بر پوشش کامل مسیرهای prime ندارد.

پاسخ:

برای این کار صرفا کافی است یک مثال نقض بیاوریم. برنامه زیر را در نظر بگیرید:



در این برنامه آزمایه $TR = \{12345, 123425\}$ تمام du path-ها را پوشش می‌دهد. با این حال مسیر 3423 که یک prime path است را پوشش نمی‌دهد. بنابراین معیار ADUPC، معیار PPC را شامل نمی‌شود.