

Q1. 撰寫客製 System call 模組 (請先做此題以防時間不夠)

檢查方式:

- 1.確認版本正確(uname -r)
- 2.執行.o 檔，確認有輸出 Call System 有回傳 0
- 3.cmd 輸入 dmesg | grep Hello 確認有正確資訊

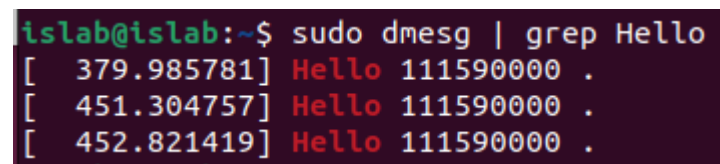
使用 ubuntu 22，下載 kernel 版本 6.4.9，使用以下程式，標註學號

```
#include <linux/kernel.h>
#include <linux/syscalls.h>
SYSCALL_DEFINE0(hello){
    printk("Hello +你的學號 .\n");
    return 0;
}
```

(1) 參考講義，進行 kernel 編譯，撰寫客製 System call

(2) 撰寫測試用.c 檔案，印出 Hello+你的學號，輸入 dmesg | grep Hello，由助教檢查完成後，截圖上傳至 Zuvio 小考二 – Q1

預期結果:



```
islalab@islalab:~$ sudo dmesg | grep Hello
[ 379.985781] Hello 111590000 .
[ 451.304757] Hello 111590000 .
[ 452.821419] Hello 111590000 .
```

Q2. Daemon 設計

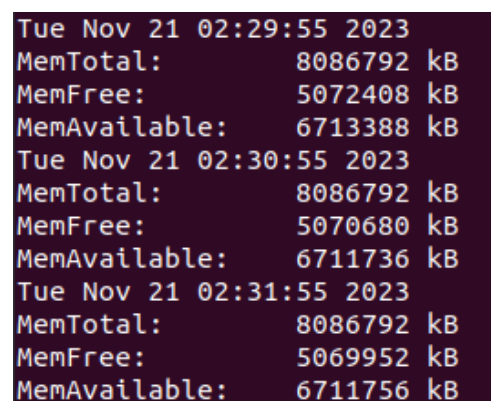
檢查方式:

請撰寫程式完成題目要求，並在檢查時解釋程式邏輯與展示程式輸出

問題描述:

寫一個 Daemon 每一分鐘記錄時間與記憶體使用資訊，利用”cat /proc/meminfo”來取得記憶體資訊，並將記錄的時間與記憶體使用資訊結果寫入一個自定義的 log 檔(time.log)，由助教檢查完成後，截圖 log 訊息畫面(至少兩分鐘以上)上傳至 Zuvio 小考二 – Q2

預期結果:



```
Tue Nov 21 02:29:55 2023
MemTotal:      8086792 kB
MemFree:       5072408 kB
MemAvailable:  6713388 kB
Tue Nov 21 02:30:55 2023
MemTotal:      8086792 kB
MemFree:       5070680 kB
MemAvailable:  6711736 kB
Tue Nov 21 02:31:55 2023
MemTotal:      8086792 kB
MemFree:       5069952 kB
MemAvailable:  6711756 kB
```

Q3. 防火牆設定 IP 連線

檢查方式:

請在 ubuntu 22.04 虛擬機開啟修改之系統檔案，確認符合題目要求

問題描述：

請開啟兩台虛擬機 DevOps、ubuntu 22.04，根據題目需求設定，進行 ssh 連線後，由助教檢查完成後，截圖防火牆過濾規則及兩個連線畫面上傳至 Zuvio 小考二 – Q3-1、Q3-2、Q3-3。

(1) 請設定 ubuntu 22.04 虛擬機允許桌機 ssh 連線

(2) 請設定 ubuntu 22.04 虛擬機不允許 DevOps 虛擬機 ssh 連線

預期結果:

使用 桌機 連線 ubuntu 22.04 虛擬機:

```
C:\Users\JB>ssh islab@192.168.190.129
islab@192.168.190.129's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.4.9 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

10 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Nov 21 15:12:14 2023 from 192.168.190.1
islab@islab:~$ |
```

使用 DevOps 虛擬機 連線 ubuntu 22.04 虛擬機:

```
islab@ubuntu:~/Desktop$ ssh 192.168.190.129
ssh: connect to host 192.168.190.129 port 22: Connection refused
```

防火牆過濾規則:

```
islab@islab:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination          tcp dpt:ftp
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp  --  192.168.190.1        anywhere             tcp dpt:ssh reject-with icmp-port-unreachable
REJECT     tcp  --  192.168.190.146      anywhere             tcp dpt:ssh reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Q4. Process fork

檢查方式:

請撰寫程式完成題目要求，並在檢查時解釋程式邏輯與展示程式輸出

問題描述:

請參考下列數學式，計算 C_n 的第 1 項至第 40 項之和

以子程序(process)計算 C_n 的第 1 項至第 15 項之和，以父程序(process)計算 C_n 的第 16 項至第 40 項之和

$$C_n = \sum_{n=1}^{\infty} \frac{n-1}{n} = 0 + \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \dots$$

(1)印出子程序第 1 項至第 15 項之和計算結果(請印出小數點後 9 位，%.9lf)

(2)父程序等待子程序結束後，印出第 16 項至第 40 項之和計算結果(請印出小數點後 9 位，%.9lf)

(3)印出父程序與子程序的總花費時間(請印出小數點後 9 位，%.9lf)

(4)印出一個程序計算上面問題的總花費時間(請印出小數點後 9 位，%.9lf)

由助教檢查完成後，將程式碼上傳至 Zuvio 小考二 – Q4

預期結果:

```
Child result:11.681771007
Parent result:24.039685954
Total time of multiprocess: 0.000155000 seconds
Total time of singleprocess: 0.000000000 seconds
```

Q5. Thread

檢查方式:

請撰寫程式完成題目要求，並在檢查時解釋程式邏輯與展示程式輸出

問題描述:

參考下列數學式，計算 B_n 的第 1 項至第 40 項之和

造出父程序和 2 個 Thread 程序，父程序計算 B_n 的第 1 項至第 10 項之和，

thread 1 計算 B_n 的第 11 項至第 25 項之和，thread 2 計算 B_n 的第 26 項至第 40 項之和。

$$B_n = \sum_{n=1}^{\infty} (-1)^{n+1} \left(\frac{1}{n}\right) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots$$

(1)印出父程序加總 3 個計算結果(請印出小數點後 9 位，%.9lf)

(2)印出父程序與 2 個 Thread 程序的總花費時間 (請印出小數點後 9 位，%.9lf)

(3)印出一個程序計算上面問題的總花費時間(請印出小數點後 9 位，%.9lf)

由助教檢查完成後，將程式碼上傳至 Zuvio 小考二 – Q5

預期結果:

```
Result: 0.680803382
Total time of multiprocess: 0.000989000 seconds
Total time of singleprocess: 0.000000000 seconds
```

Q6. 共享記憶體

檢查方式:

請撰寫程式完成題目要求，並在檢查時解釋程式邏輯與展示程式輸出

問題描述:

請參考下列數學式，以子程序(process)計算 e 的第 0 項至第 2 項之和，並以父程序(process)計算 e 的第 3 項至第 10 項之和

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1*2} + \frac{1}{1*2*3} + \dots$$

(1)子程序印出自己第 0 項至第 2 項之和計算結果(請印出小數點後 9 位，

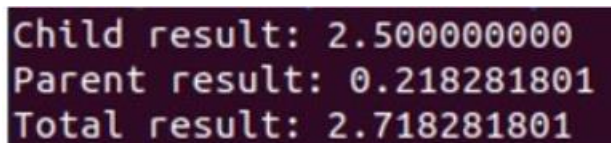
% .9lf)，並透過共享記憶體將自己的計算結果傳遞給父程序

(2)父程序等待子程序結束後，印出自己第 3 項至第 10 項之和計算結果，接著再列印出與子程序的計算結果相加後的計算結果(請印出小數點後 9 位，

% .9lf)

由助教檢查完成後，將程式碼上傳至 Zuvio 小考二 – Q6

預期結果:



```
Child result: 2.500000000
Parent result: 0.218281801
Total result: 2.718281801
```

Q7. Jenkins & Unit Test & Coverage

(1) Jenkins

檢查方式:

請確認 Jenkins 專案中有無 Dog 之 class，並可在 Jacoco Report 中找到

問題描述:

請參考 DevOps 講義架設 Jenkins，並在專案中的 production code 資料夾中新增 Dog.java，在 test code 資料夾中新增 DogTest.java，commit & push 並觸發 Jenkins 建置後，進入 JaCoCo Coverage Report 中的 main.java.example.dog 頁面，由助教檢查完成後，截圖並上傳至 Zuvio 小考二 – Q7-1、Q7-2。

Dog.java:

```
package main.java.example;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
public class Dog {
```

```
    private String name;
```

```
    private Double weight;
```

```
private String sex;
private Integer age;
private List<String> tricks;

public Dog(String name, Double weight, String sex, Integer age) {
    this.name = name;
    this.weight = weight;
    this.sex = sex;
    this.age = age;
    this.tricks = new ArrayList<>();
}

public String getName() {
    return "My name is " + name + ".";
}

public String getItsHobby() {
    if (sex == "boy")
        return "Bite something.";
    else
        return "Play Ball.";
}

public void teachTrick(String trick) {
    tricks.add(trick);
}

public String showTricks() {
    String result = "";
    if (tricks.isEmpty()) {
        result = name + " have no tricks.";
    } else {
        result += name + " have tricks:";
        for (String trick : tricks) {
            result += " " + trick;
        }
    }
    return result;
}
```

```

    }

    public Integer getHowFastCanItRun() {
        if(age < 1 || age > 10)
            return 1;
        else if (weight >= 5 && weight <= 10)
            return 5;
        else
            return 3;
    }
}

```

DogTest.java:

```
package test.java.example;
```

```
import org.junit.Assert;
import org.junit.Test;
import main.java.example.Dog;

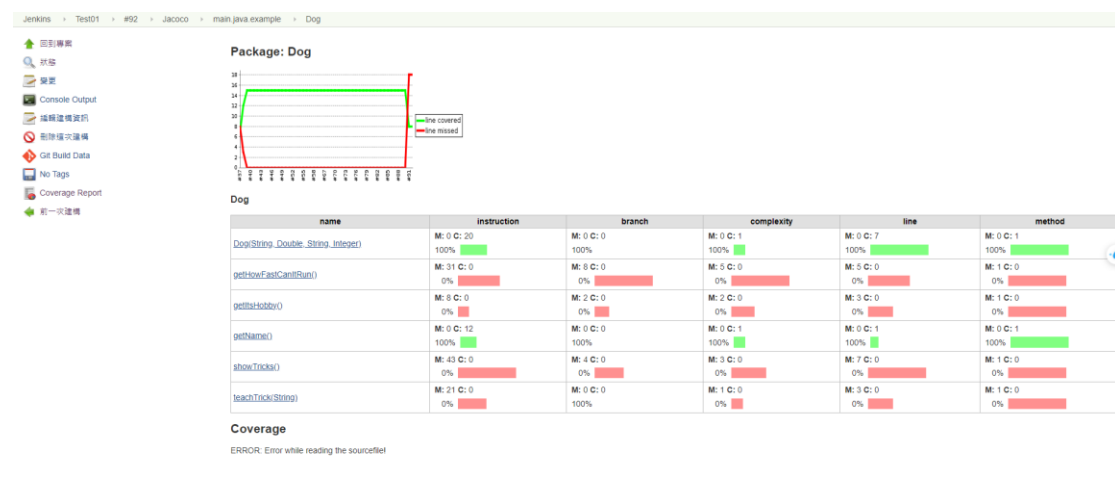
```

```

public class DogTest {
    @Test
    public void testGetName() {
        Dog dog = new Dog("doggy", 8.5, "boy", 3);
        Assert.assertEquals("My name is doggy.", dog.getName());
    }
}

```

預期結果:



(2) Unit Test & Coverage

檢查方式:

請開啟 Jacoco Coverage Report，確認 Dog 之 Coverage 是否 100%。

問題描述:

接續(1)，完成 Dog.java 的 Unit Test，使其 Line Coverage & Branch Coverage 達到 100%，觸發 Jenkins 並成功建置後，進入 Jacoco Coverage Report 中的 main.java.example.dog，由助教檢查完成後，截圖並上傳至 Zuvio 小考二 – Q7-2

Q8. SonarQube 掃描及重構程式碼

(1) SonarQube 掃描

檢查方式:

參考 DevOps 講義使用 DevOps 虛擬機，使用講義中 git 專案 proj02，並在專案新增提供的程式碼(SimpleCodeExample.java)以進行重構並 commit，使用 Jenkins 建置專案後，查看 SonarQube 報告。

問題描述:

請將以下的程式碼存成 SimpleCodeExample.java，使用 Jenkins 建置後查看 SonarQube，會顯示存在 4 個 Code Smells，截圖上傳至 Zuvio 小考二 – Q8-1。

SimpleCodeExample.java:

```
package main.java.example;
```

```
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
```

```
import java.util.ArrayList;
```

```
public class SimpleCodeExample {
```

```
    Logger logger = Logger.getLogger(SimpleCodeExample.class.getName());
```

```
    public void upperCase(String str) {
```

```
        if (logger.isLoggable(Level.INFO) && str.length() > 0) {
```

```
            logger.log(Level.INFO, str.toUpperCase());
```

```
        }
```

```
    }
```

```
    public void unusedParameter(int x, int y) {
```

```
        logger.log(Level.INFO, "X: {0}", x);
```

```
        int z = 42;
```

```

    }

    public static void main(String[] args) {
        SimpleCodeExample example = new SimpleCodeExample();
        example.upperCase("test");
        example.unusedParameter(10, 20);
    }
}

```

預期結果:

The screenshot shows the SonarQube interface with the 'Code Smell' filter selected. The main panel displays a list of issues for the file 'src/main/java/example/SimpleCodeExample.java'. The issues are:

- Remove this unused import 'java.util.ArrayList'. (Code Smell, Minor, Open, Not assigned, 2min effort)
- Remove this unused method parameter 'y'. (Code Smell, Major, Open, Not assigned, 5min effort)
- Remove this useless assignment to local variable 'z'. (Code Smell, Major, Open, Not assigned, 15min effort)
- Remove this unused 'z' local variable. (Code Smell, Minor, Open, Not assigned, 5min effort)

(2) SonarQube 掃描

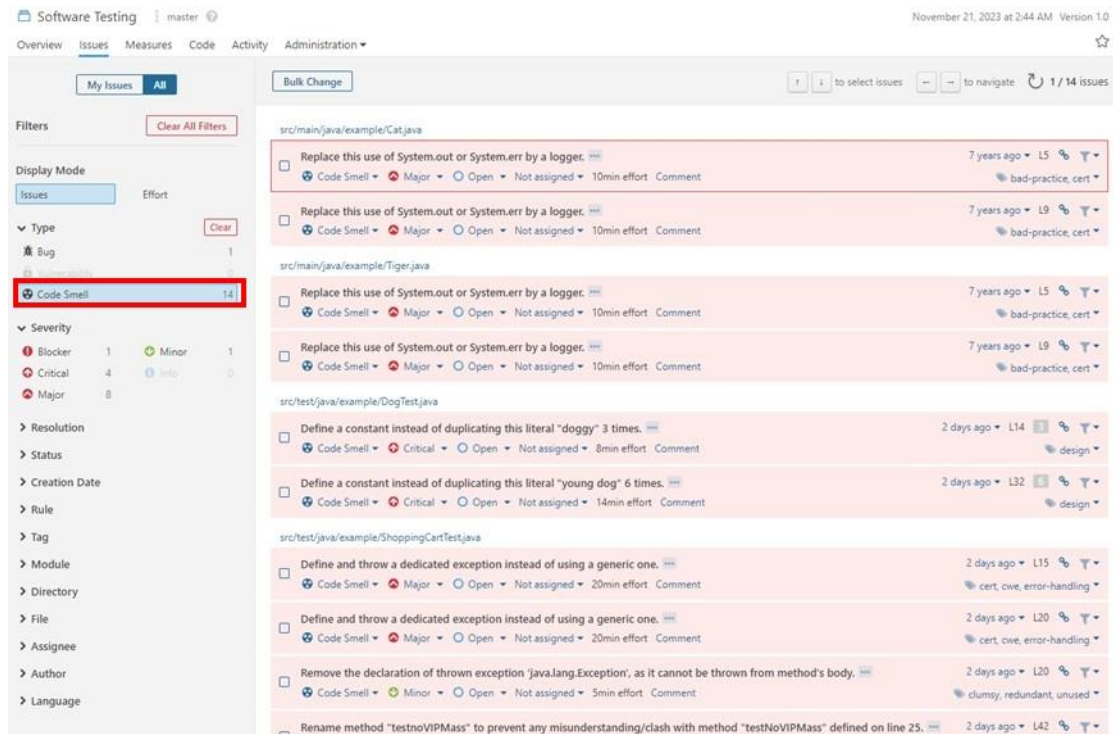
檢查方式:

請開啟 SonarQube 報告，確認 SimpleCodeExample 沒有存在 Code Smells

問題描述:

接續(1)，請將 SimpleCodeExample.java 進行重構，將 Code Smells 移除並 commit 後，使用 Jenkins 建置後，查看 SonarQube SonarQube 報告，確認 Code Smells 是否有被移除，截圖上傳至 Zuvio 小考二 – Q8-2

。



Q9. JMeter

檢查方式:

請在 JMeter 點選 Result Tree，確認有無大部分 Request 都回傳 200 (綠色)，並確認各小題的需求是否完成

問題描述:

請參考 JMeter 講義，並針對 Calculator.jsp 頁面進行效能測試

(1) 請設計 Calculator.jsp 頁面之測試腳本，並能成功執行，測試完成並由助教檢查完成後截圖 Summary Report 上傳至 Zuvio 小考二 - Q9-1

(測試需包含進入 Calculator.jsp 頁面、選擇計算模式以及填入兩個運算元欄位、點擊 calculate 按鈕)

(測試腳本可自行撰寫或透過 Proxy 錄製)

(測試結果需提供 Summary Report、Result Tree、Aggregate Graph)

(測試腳本之執行緒和迴圈請設定為 10)

Calculator.jsp:

```
<%@ page contentType="text/html; charset=big5" language="java"
import="java.sql.*" errorPage="" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=big5">
<title>Calculator</title>
</head>
```

```
<body>
<h2>Simple Calculator</h2>
<%
```

```
String strSub = request.getParameter("submit");
```

```
if(strSub == null){
    %>
```

```
<form name="calculator" method="post" action="Calculator.jsp">
```

```
    Mode:
```

```
    <select name="mode">
```

```
        <option value="+" selected>+</option>
```

```
        <option value="-">-</option>
```

```
        <option value="*">*</option>
```

```
        <option value="/">/</option>
```

```
    </select><br />
```

```
    <input type="text" name="Field1">
```

```
    <input type="text" name="Field2">
```

```
    <input type="submit" name="submit" value="calculate">
```

```
</form>
```

```
<%
```

```
}else{
```

```
int sum = 0;
```

```
int field1 = Integer.parseInt(request.getParameter("Field1"));
```

```
int field2 = Integer.parseInt(request.getParameter("Field2"));
```

```
String mode = request.getParameter("mode");
```

```
switch (mode) {
```

```
    case "+":
```

```
        sum = field1 + field2;
```

```
        break;
```

```
    case "-":
```

```
        sum = field1 - field2;
```

```
        break;
```

```
    case "*":
```

```

        sum = field1 * field2;
        break;
    case "/":
        sum = field1 / field2;
        break;
}
out.print("The calculation = "+ sum+"<P><a href='Calculator.jsp'>restart to
calculate!</a>");
}
%>
</body>
</html>

```

calculate_data.csv:

45,+4

20,-7

80,*2

100,/4



Simple Calculator

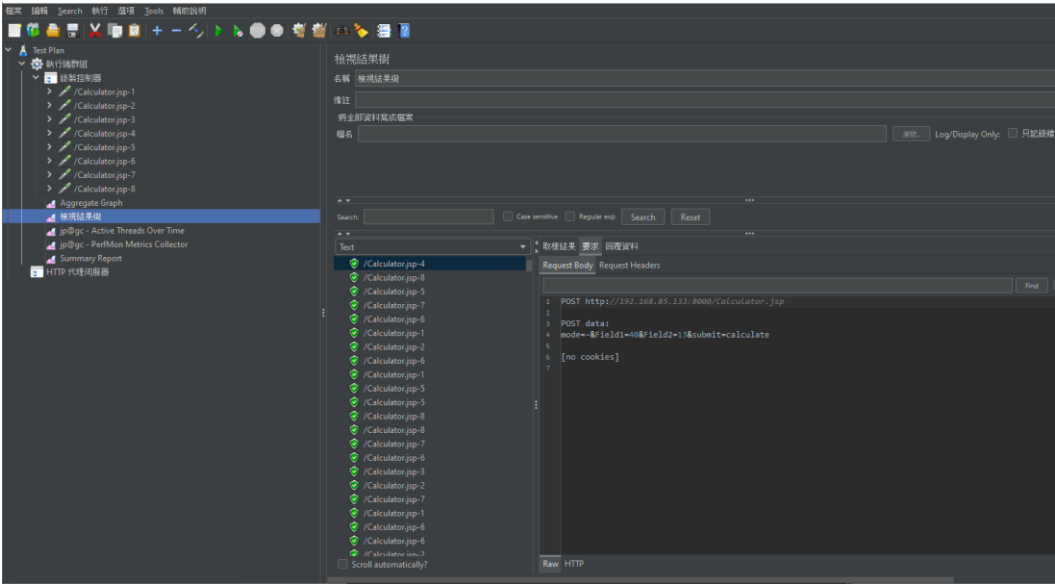
Mode: + ▼

預期結果:

Summary Report:

Summary Report										
名稱: Summary Report										
備註:										
將全部資料寫成檔案										
檔名: <input type="text"/> <input type="button" value="瀏覽..."/> <input type="checkbox"/> Log/Display Only: <input type="checkbox"/> 只監錄錯誤 <input type="checkbox"/> Suc										
Label	取樣數	平均值	最小值	最大值	Std. Dev.	拋球率	處理量	每秒佇位泥組	Sent KB/sec	
/Calculator.jsp-1	100	30	0	53	8.77	0.00%	31.8/sec	27.11	14.38	
/Calculator.jsp-2	100	29	0	54	9.40	0.00%	32.2/sec	17.13	21.91	
/Calculator.jsp-3	100	29	0	48	8.99	0.00%	32.3/sec	27.52	16.44	
/Calculator.jsp-4	100	29	0	51	8.63	0.00%	32.4/sec	17.26	22.05	
/Calculator.jsp-5	100	29	0	53	8.90	0.00%	32.5/sec	27.70	16.55	
/Calculator.jsp-6	100	29	0	53	8.91	0.00%	32.6/sec	17.41	22.16	
/Calculator.jsp-7	100	29	0	44	8.46	0.00%	32.7/sec	27.87	16.65	
/Calculator.jsp-8	100	29	0	53	10.06	0.00%	32.8/sec	17.44	22.34	
總計	800	29	0	54	8.99	0.00%	250.3/sec	173.22	147.22	

Result Tree:



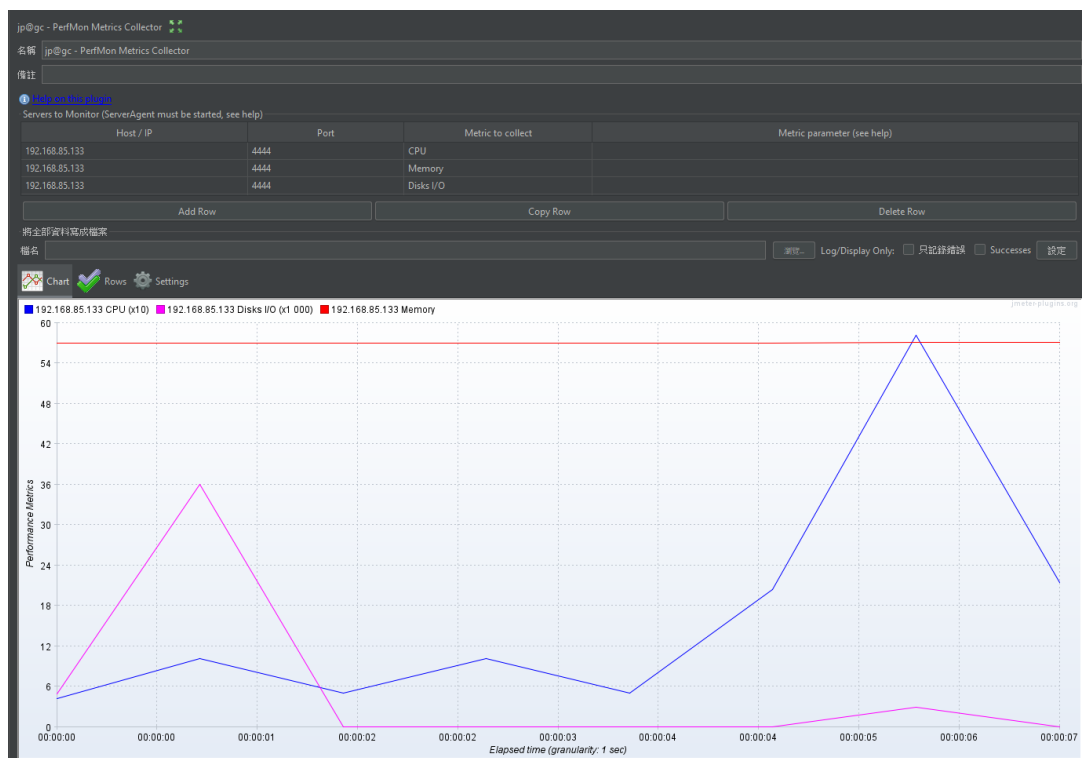
(2) 使用提供之 CSV 檔案，並修改(1)之測試案例，使其能夠讀取 CSV 檔案中的兩個運算元及一個運算子，並能成功執行，測試完成並由助教檢查完成後，截圖 Aggregate Graph 上傳至 Zuvio 小考二 – Q9-2

預期結果:

Aggregate Graph													
名稱 Aggregate Graph													
備註													
將全部資料寫成檔案													
圖名													
Log/Display Only: <input type="checkbox"/> 只記錄錯誤 <input type="checkbox"/> Successes <input type="checkbox"/> 設定													
Label	取樣數	平均值	中間值	90% Line	95% Line	99% Line	最小值	最大值	錯誤率	處理量	每秒千位元組	Sent KB/sec	
/Calculator.jsp-1	100	26	27	37	39	46	7	46	0.00%	35.2/sec	30.00	16.13	
/Calculator.jsp-2	100	26	27	38	39	42	6	46	0.00%	35.5/sec	18.91	24.15	
/Calculator.jsp-3	100	25	26	37	39	40	7	46	0.00%	35.7/sec	30.37	18.15	
/Calculator.jsp-4	100	25	26	38	39	41	7	41	0.00%	35.7/sec	19.01	24.28	
/Calculator.jsp-5	100	25	25	37	39	41	6	41	0.00%	35.8/sec	30.49	18.22	
/Calculator.jsp-6	100	26	27	38	38	41	7	46	0.00%	35.9/sec	19.09	24.38	
/Calculator.jsp-7	100	25	25	38	39	46	6	46	0.00%	35.9/sec	30.60	18.28	
/Calculator.jsp-8	100	25	27	37	38	39	5	41	0.00%	36.0/sec	19.16	24.48	

(3) 接續(2)，啟動監控程式、JMeter 啟動 PerfMon 監聽功能，監聽 CPU、Memory、Disks I/O 三種資源，並能成功執行(2)，測試完成並由助教檢查完成後，截圖 PerfMon Metrics Collector 上傳至 Zuvio 小考二 – Q9-3

預期結果:



Q10. Zuvio 心得

簡要說明本學期在課程學到之技術及感想，並上傳至 Zuvio 小考二 – Q10 。
(60 字有意義的句子才計分)