Coursework Header Sheet

241360-11

UNIVERSITY
of
GREENWICH

Course	COMP1562: Operating Systems	Course School/Level	H/UG
Coursework	COMP1562 Coursework	Assessment Weight	40.00%
Tutor	M Pelc	Submission Deadline	11/03/2019

Logbook comprising weekly uploads

Coursework is receipted on the understanding that it is the student's own work and that it has not, in whole or part, been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged in accordance with the University's Regulations regarding Cheating and Plagiarism.

001001720

Tutor's comments

Grade Awarded__________ ***For Office Use Only***__________ ***Final Grade***__________

Moderation required: *yes/no* ***Tutor***__________ ***Date*** __________

Table of Contents

COMP1562 Logbook (Week 1)

Basic Information

1.1	Student name	**001001720**
1.2	Who did you work with? Name and/or id	**001006731**
1.3	Which lab topic does this document relate to?	System Bit Processors and System Shells
1.4	How well do you feel you have done?	I have completed the exercise and am totally satisfied with my work.
1.5	Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete the tasks. Proof of that is shown below.

Task 1:

1.1:

In this task, we were asked to use the fetch-execute cycle to replicate pseudo program execution by loading, adding and storing; three tasks that are normally executed within a CPU. The specifications of the machine are shown below:

Hypothetical 16 bit processor

16 bit accumulator AC
16 bit instruction register IR
12 bit program counter PC

Instruction format:
4 bit opcode
12 bit address

opcodes:

0011 = Load AC from I/O **3**

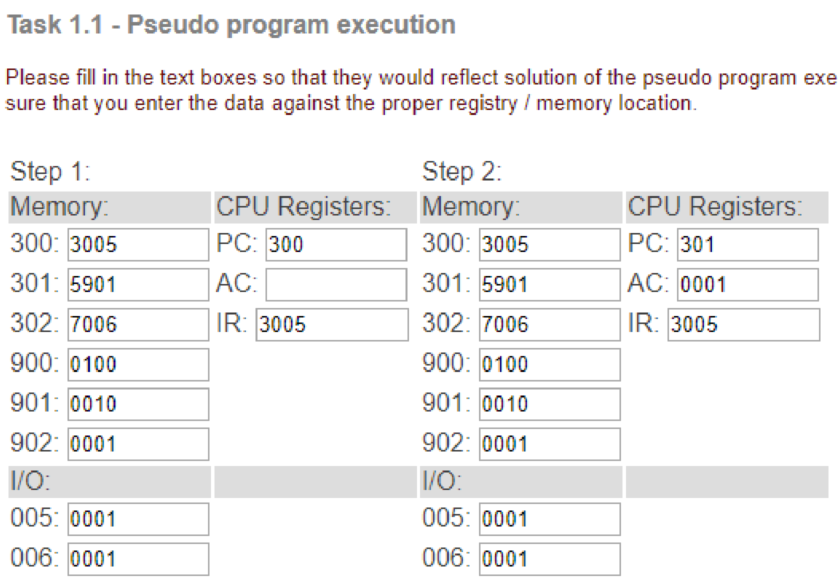
0101 = Add to AC (add date from the given memory location) **5**

0111 = Store AC to I/O **7**

Annotated screen shots demonstrating what you have achieved:

Screenshots showing my correct answers for task 1.1:

Steps 1 and 2 showing the task of loading AC from the device buffer:

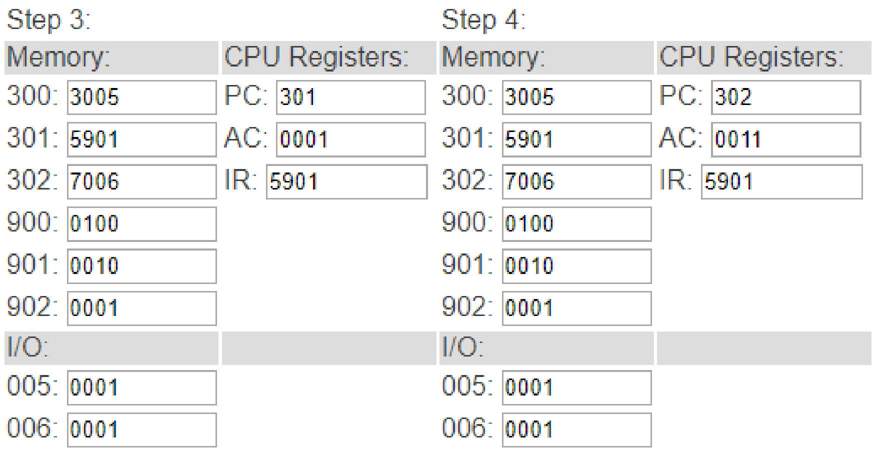


Task 1.1 - Pseudo program execution

Please fill in the text boxes so that they would reflect solution of the pseudo program execution task. Make sure that you enter the data against the proper registry / memory location.

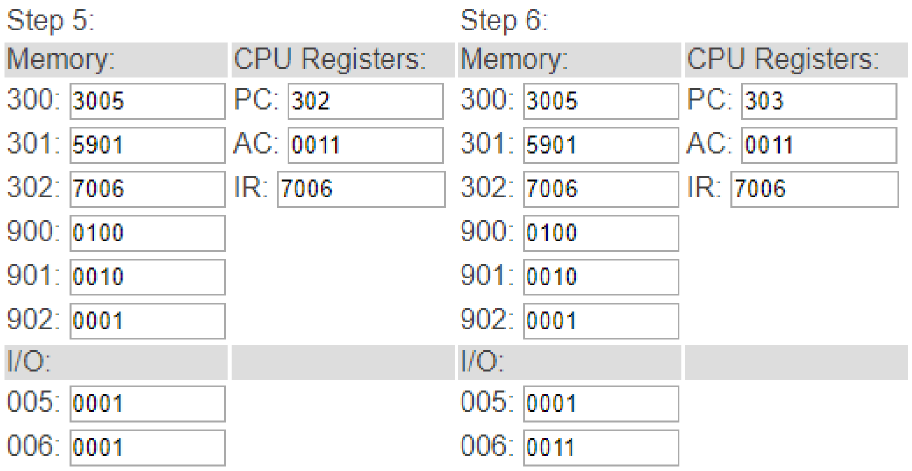
Step 1:				Step 2:			
Memory:		CPU Registers:		Memory:		CPU Registers:	
300:	3005	PC:	300	300:	3005	PC:	301
301:	5901	AC:		301:	5901	AC:	0001
302:	7006	IR:	3005	302:	7006	IR:	3005
900:	0100			900:	0100		
901:	0010			901:	0010		
902:	0001			902:	0001		
I/O:				I/O:			
005:	0001			005:	0001		
006:	0001			006:	0001		

Steps 3 and 4 showing the task of adding info to the next memory location:



Step 3:				Step 4:			
Memory:		CPU Registers:		Memory:		CPU Registers:	
300:	3005	PC:	301	300:	3005	PC:	302
301:	5901	AC:	0001	301:	5901	AC:	0011
302:	7006	IR:	5901	302:	7006	IR:	5901
900:	0100			900:	0100		
901:	0010			901:	0010		
902:	0001			902:	0001		
I/O:				I/O:			
005:	0001			005:	0001		
006:	0001			006:	0001		

Steps 5 and 6 showing the task of storing the info from AC into the device buffer:



Step 5:				Step 6:			
Memory:		CPU Registers:		Memory:		CPU Registers:	
300:	3005	PC:	302	300:	3005	PC:	303
301:	5901	AC:	0011	301:	5901	AC:	0011
302:	7006	IR:	7006	302:	7006	IR:	7006
900:	0100			900:	0100		
901:	0010			901:	0010		
902:	0001			902:	0001		
I/O:				I/O:			
005:	0001			005:	0001		
006:	0001			006:	0011		

1.2:

In this task, we were asked to calculate the cache hit ratio using the machine specifications shown below:

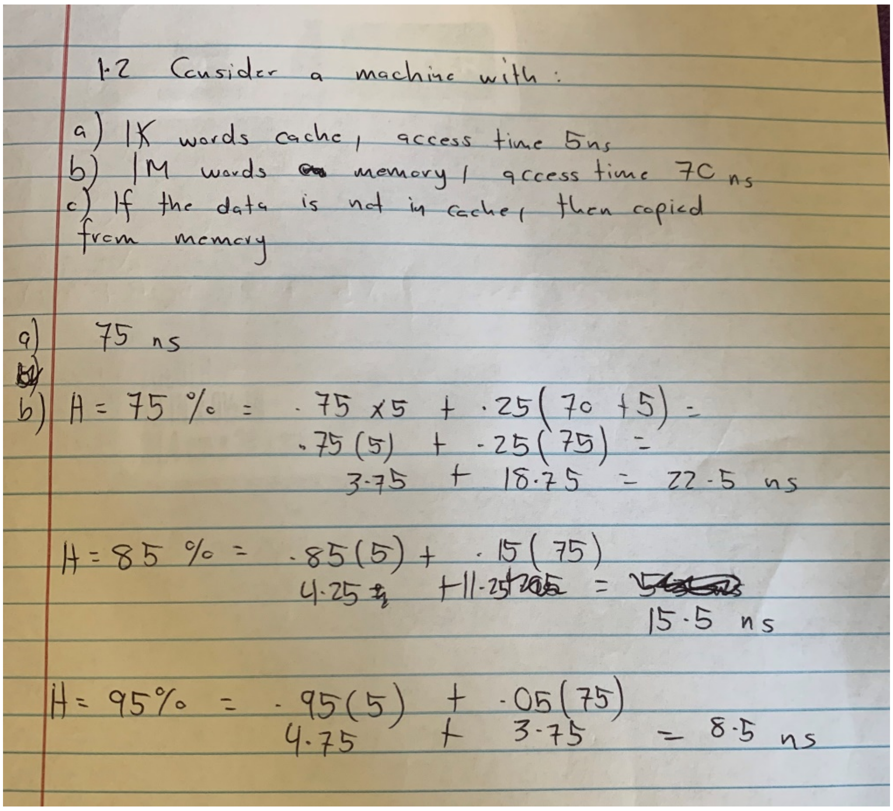
A) 1k Words Cache, Access Time 5ns

B) 1M words memory, Access time 70ns

C) If the data is not in the Cache, then it is copied from the memory

Annotated screen shots demonstrating what you have achieved:

My working out shown on paper:



1.2 Consider a machine with:

a) 1K words cache, access time 5ns
b) 1M words memory, access time 70 ns
c) If the data is not in cache, then copied from memory

a) 75 ns

b) H = 75% = .75 x5 + .25(70 +5) =
.75 (5) + .25(75) =
3.75 + 18.75 = 22.5 ns

H = 85% = .85(5) + .15(75)
4.25 + 11.25 = 15.5 ns

H = 95% = .95(5) + .05(75)
4.75 + 3.75 = 8.5 ns

Calculations:

a) Cache miss time: (70 ns + 5 ns) = **75 ns**

b) H= 75% = .75 x 5 + .25 (70 + 5) =

.75 x 5 + .25 (75) =

3.75 + 18.75 = **22.5 ns**

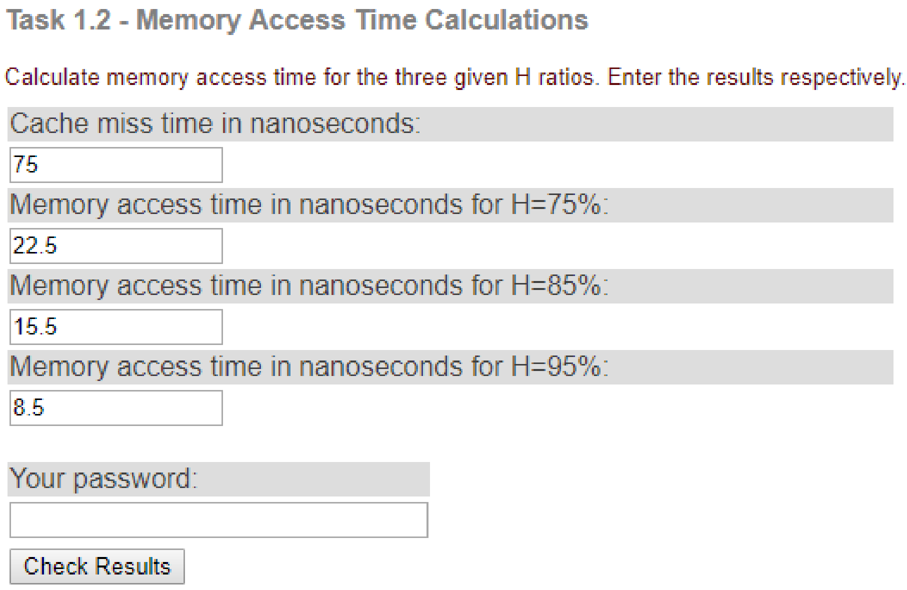
H= 85% = .85 x 5 + .15 (70 + 5) =

4.25 + 11.25 = **15.5 ns**

H= 95% = .95 x 5 + .05 (70 + 5) =

4.75 + 3.75 = **8.5 ns**

Screenshot showing correct answers for task 1.2:



Task 1.2 - Memory Access Time Calculations

Calculate memory access time for the three given H ratios. Enter the results respectively.

Cache miss time in nanoseconds: 75

Memory access time in nanoseconds for H=75%: 22.5

Memory access time in nanoseconds for H=85%: 15.5

Memory access time in nanoseconds for H=95%: 8.5

Your password:

Personal Reflection:

I learned a lot about system bit processors and the different types of storage during this week's lecture and lab sessions. This has definitely increased my knowledge about the way operating systems work, as I did not know some of this information beforehand. Learning about the different types of architectures was interesting, but not as helpful for the lab exercises. The easiest and most interesting part of my learning was the bit about calculating the cache hit ratio, as well as finding out that it is one of the fastest types of memory only behind registers. I was able to successfully calculate and finish the problems presented in task 1.2. The most challenging aspect I encountered was the pseudo-program execution. I was already familiar with the fetch execute cycle but found it difficult to integrate into the problem. However, I was not that familiar with the Von Neumann system, and it took me a few days to fully acclimatize to the requirements outside of the lecture slides. With the help of my group members and through trial and error, I was able to complete the task and I do feel like my understanding of operating systems has increased.

COMP1562 Logbook (Week 2)

Basic Information

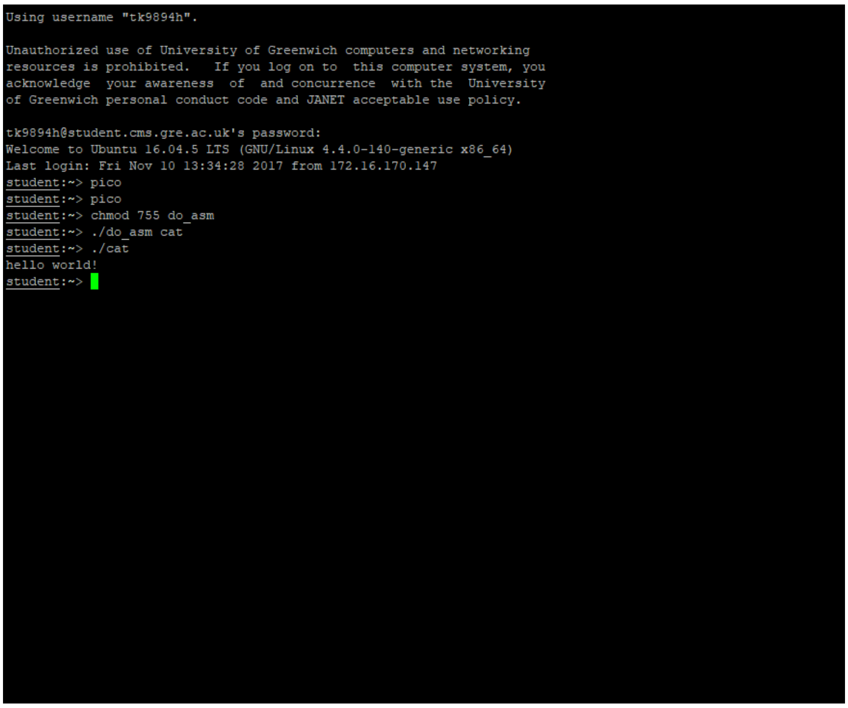
1.1 Student name	**001001720**
1.2 Who did you work with? Name and/or id	**001006731**
1.3 Which lab topic does this document relate to?	System Bit Processors and System Shells
1.4 How well do you feel you have done?	I have completed the exercise and am totally satisfied with my work.
1.5 Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete the tasks. Proof of that is shown below.

Annotated screen shots demonstrating what you have achieved:

TASK 1:

Exercise 1:

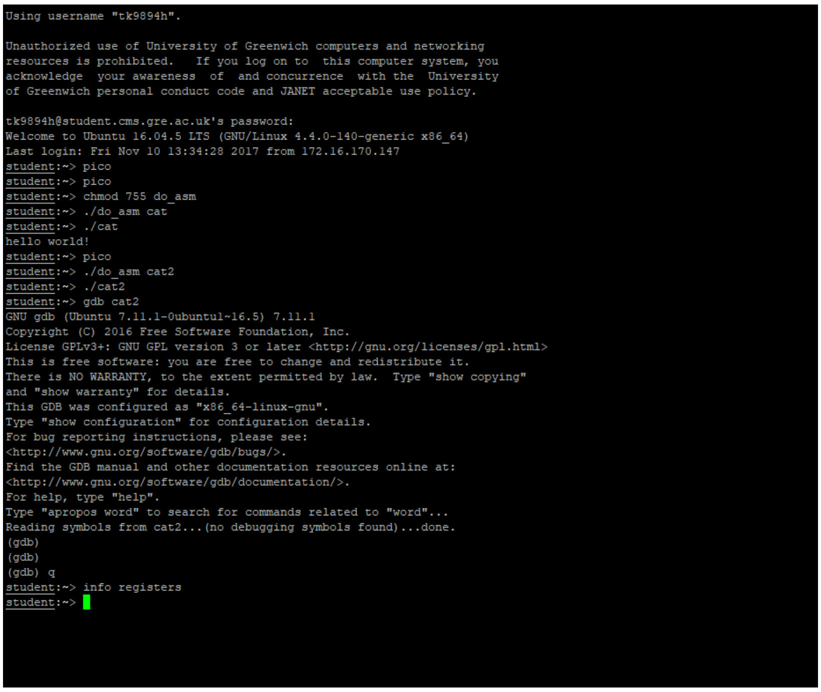
Screenshot showing successful completion of exercise 1, which was to run the code to display "Hello World". The saved program is named 'cat':



```
Using username "tk9894h".

Unauthorized use of University of Greenwich computers and networking
resources is prohibited.   If you log on to  this computer system, you
acknowledge  your awareness  of  and concurrence  with the  University
of Greenwich personal conduct code and JANET acceptable use policy.

tk9894h@student.cms.gre.ac.uk's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-140-generic x86_64)
Last login: Fri Nov 10 13:34:28 2017 from 172.16.170.147
student:~> pico
student:~> pico
student:~> chmod 755 do_asm
student:~> ./do_asm cat
student:~> ./cat
hello world!
student:~>
```

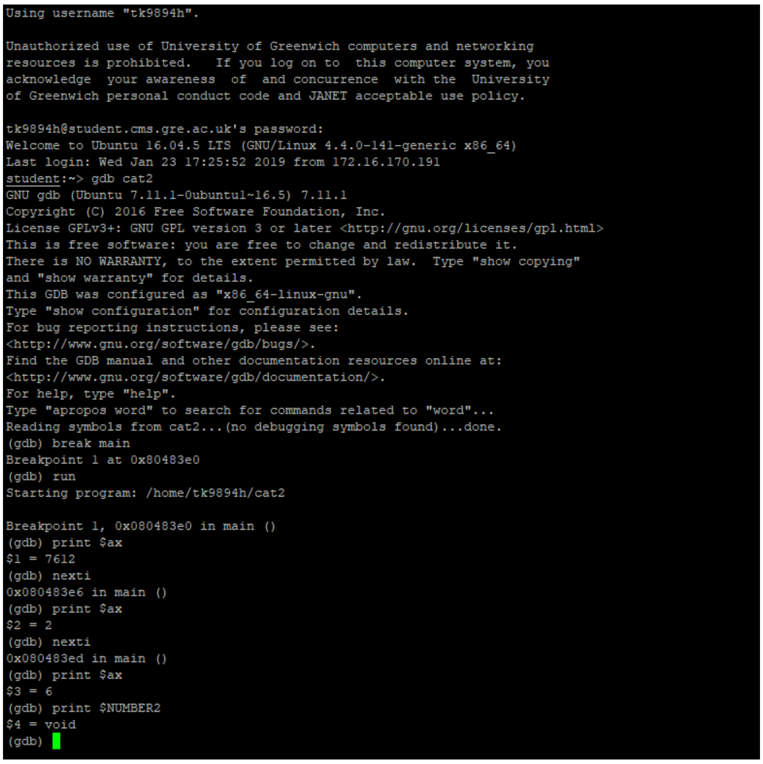
Exercise 2:

This exercise familiarized us with the use of the debugger.
Shown below is the creation of the second program named ‘cat2’:



```
Using username "tk9894h".

Unauthorized use of University of Greenwich computers and networking
resources is prohibited.   If you log on to  this computer system, you
acknowledge  your awareness  of  and concurrence  with the  University
of Greenwich personal conduct code and JANET acceptable use policy.

tk9894h@student.cms.gre.ac.uk's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-140-generic x86_64)
Last login: Fri Nov 10 13:34:28 2017 from 172.16.170.147
student:~> pico
student:~> pico
student:~> chmod 755 do_asm
student:~> ./do_asm cat
student:~> ./cat
hello world!
student:~> pico
student:~> ./do_asm cat2
student:~> ./cat2
student:~> gdb cat2
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from cat2...(no debugging symbols found)...done.
(gdb)
(gdb)
(gdb) q
student:~> info registers
student:~>
```

After debugging, shown below is the use of the ‘nexti’ command to step through the program and ensure that the correct values are in each register. The final answer is 6, shown in register $3:



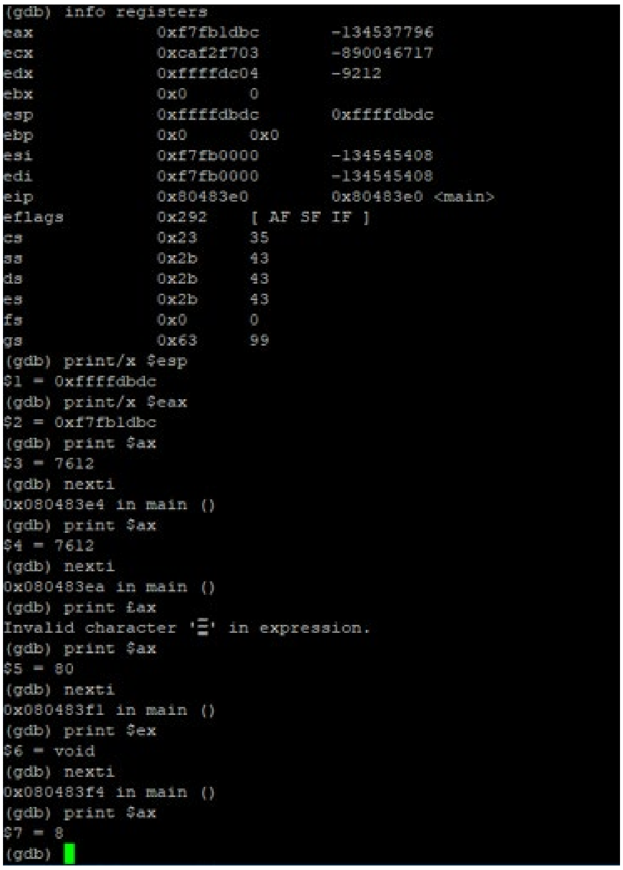
```
Using username "tk9894h".

Unauthorized use of University of Greenwich computers and networking
resources is prohibited.   If you log on to  this computer system, you
acknowledge  your awareness  of  and concurrence  with the  University
of Greenwich personal conduct code and JANET acceptable use policy.

tk9894h@student.cms.gre.ac.uk's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-141-generic x86_64)
Last login: Wed Jan 23 17:25:52 2019 from 172.16.170.191
student:~> gdb cat2
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from cat2...(no debugging symbols found)...done.
(gdb) break main
Breakpoint 1 at 0x80483e0
(gdb) run
Starting program: /home/tk9894h/cat2

Breakpoint 1, 0x080483e0 in main ()
(gdb) print $ax
$1 = 7612
(gdb) nexti
0x080483e6 in main ()
(gdb) print $ax
$2 = 2
(gdb) nexti
0x080483ed in main ()
(gdb) print $ax
$3 = 6
(gdb) print $NUMBER2
$4 = void
(gdb)
```

Exercise 3:

Shown below is the modified program to change to the division of two numbers:
Code:

```
section .data
NUMBER1:            dw 80
NUMBER2:            dw 10

section .text
        global main

main:

        mov dx,0
        mov ax,[NUMBER1]
        mov cx,[NUMBER2]
        div cx

        mov eax,1
        mov ebx,0
        int 80h
```

After debugging, shown below is the use of the ‘nexti’ command to step through the program and ensure that the correct values are in each register. The final answer is 8, shown in register $7:



```
(gdb) info registers
eax            0xf7fb1dbc       -134537796
ecx            0xcaf2f703       -890046717
edx            0xffffdc04       -9212
ebx            0x0      0
esp            0xffffdbdc       0xffffdbdc
ebp            0x0      0x0
esi            0xf7fb0000       -134545408
edi            0xf7fb0000       -134545408
eip            0x80483e0        0x80483e0 <main>
eflags         0x292    [ AF SF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x63     99
(gdb) print/x $esp
$1 = 0xffffdbdc
(gdb) print/x $eax
$2 = 0xf7fb1dbc
(gdb) print $ax
$3 = 7612
(gdb) nexti
0x080483e4 in main ()
(gdb) print $ax
$4 = 7612
(gdb) nexti
0x080483ea in main ()
(gdb) print £ax
Invalid character '☰' in expression.
(gdb) print $ax
$5 = 80
(gdb) nexti
0x080483f1 in main ()
(gdb) print $ex
$6 = void
(gdb) nexti
0x080483f4 in main ()
(gdb) print $ax
$7 = 8
(gdb) 
```

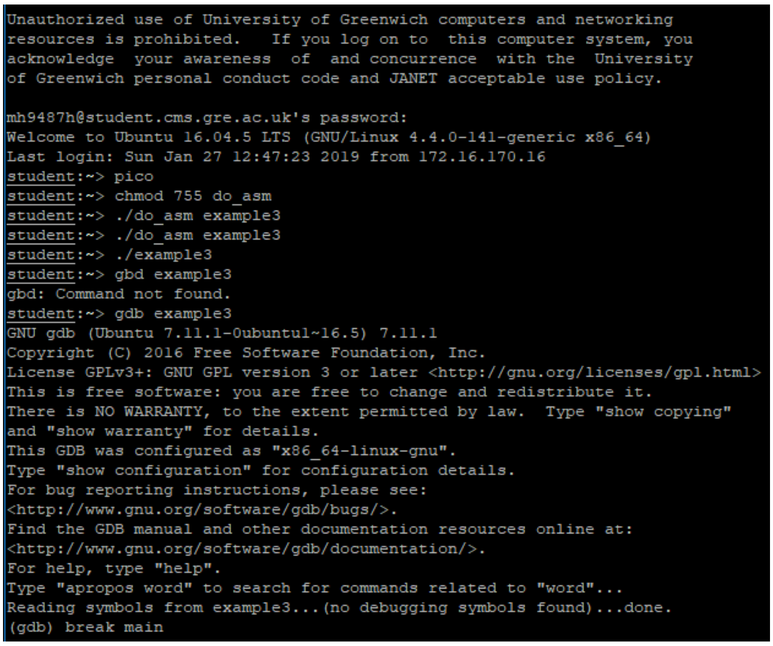
TASK 2:

This task asked us to produce a program that can calculate the area of a trapezoid.

$$\boldsymbol{result} = \frac{\boldsymbol{a}+\boldsymbol{b}}{\boldsymbol{2}} \times \boldsymbol{h}$$

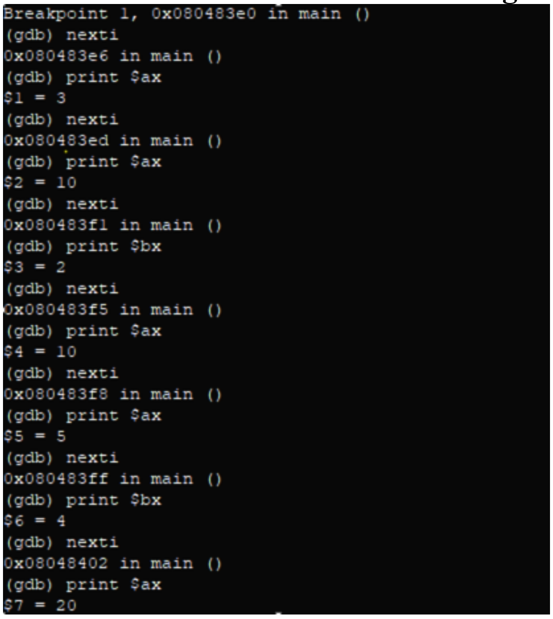
Expected Result:
= 3+7/2 *4
=5*4
=**20**

Shown below is the creation of the program, as well as running the debugger:



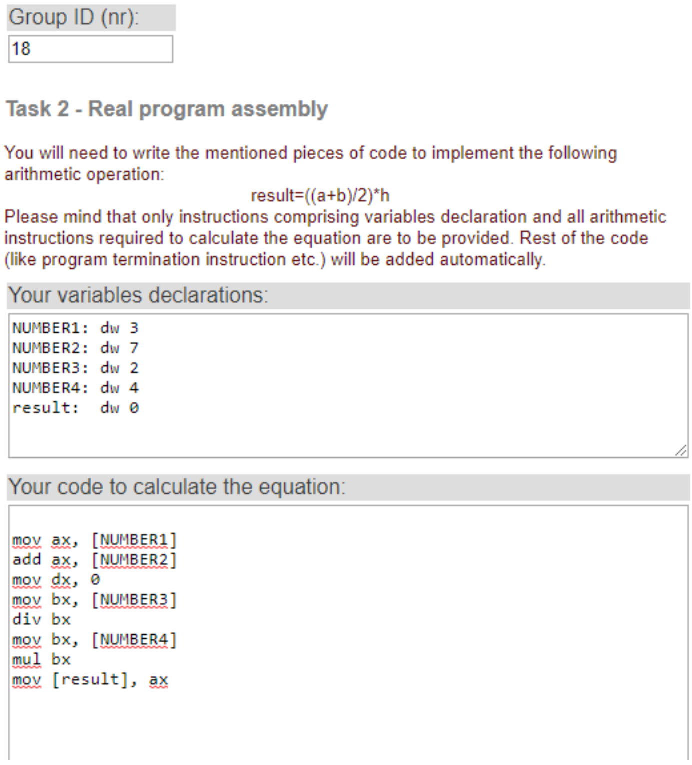
```
Unauthorized use of University of Greenwich computers and networking
resources is prohibited.   If you log on to  this computer system, you
acknowledge  your awareness  of  and concurrence  with the  University
of Greenwich personal conduct code and JANET acceptable use policy.

mh9487h@student.cms.gre.ac.uk's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-141-generic x86_64)
Last login: Sun Jan 27 12:47:23 2019 from 172.16.170.16
student:~> pico
student:~> chmod 755 do_asm
student:~> ./do_asm example3
student:~> ./do_asm example3
student:~> ./example3
student:~> gbd example3
gbd: Command not found.
student:~> gdb example3
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from example3...(no debugging symbols found)...done.
(gdb) break main
```

After debugging, shown below is the use of the 'nexti' command to step through the program and ensure that the correct values are in each register. The final answer is 20, shown in register $7:



```
Breakpoint 1, 0x080483e0 in main ()
(gdb) nexti
0x080483e6 in main ()
(gdb) print $ax
$1 = 3
(gdb) nexti
0x080483ed in main ()
(gdb) print $ax
$2 = 10
(gdb) nexti
0x080483f1 in main ()
(gdb) print $bx
$3 = 2
(gdb) nexti
0x080483f5 in main ()
(gdb) print $ax
$4 = 10
(gdb) nexti
0x080483f8 in main ()
(gdb) print $ax
$5 = 5
(gdb) nexti
0x080483ff in main ()
(gdb) print $bx
$6 = 4
(gdb) nexti
0x08048402 in main ()
(gdb) print $ax
$7 = 20
```

Shown below is the code we used to calculate the answer:



Group ID (nr):

18

Task 2 - Real program assembly

You will need to write the mentioned pieces of code to implement the following arithmetic operation:

result=((a+b)/2)*h

Please mind that only instructions comprising variables declaration and all arithmetic instructions required to calculate the equation are to be provided. Rest of the code (like program termination instruction etc.) will be added automatically.

Your variables declarations:

```
NUMBER1: dw 3
NUMBER2: dw 7
NUMBER3: dw 2
NUMBER4: dw 4
result:  dw 0
```

Your code to calculate the equation:

```
mov ax, [NUMBER1]
add ax, [NUMBER2]
mov dx, 0
mov bx, [NUMBER3]
div bx
mov bx, [NUMBER4]
mul bx
mov [result], ax
```

Personal Reflection:

This weeks' lab was slightly more challenging than last weeks', but it was still very enjoyable to complete. The lab required us to use the 'Putty' application, one that we had not used since first year, so it took us a couple of days to find our old passwords. We were already familiar with pseudo program execution from the previous weeks' task, so we were able to jump in right away. The toughest part of this lab was not reading the instructions carefully. Fetching instructions from the next register is done by using the 'nexti' command. However, I kept accidentally using the 'next' command, which moves on to the next breakpoint. This kept causing the program to terminate as we only had one breakpoint. We also had problems executing the Task 2 assembly program at first, as it was running in scriptcheck and made sense logically on paper. In the end, we were able to follow the previous examples again and apply it to finally reach the correct solution. I am glad that I experienced this lab session and 100% believe that is has made me a better programmer and given me a better understanding of system shells.

COMP1562 Logbook (Week 3)

Basic Information

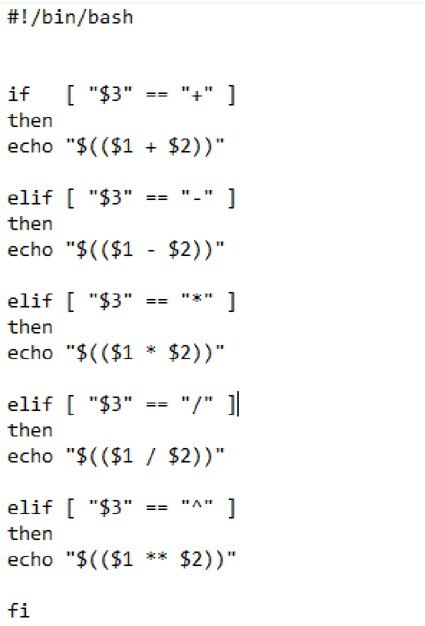
1.1 Student name	**001001720**
1.2 Who did you work with? Name and/or id	**001006731**
1.3 Which lab topic does this document relate to?	Shell Programming
1.4 How well do you feel you have done?	I have completed the exercise and am totally satisfied with my work.
1.5 Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete the tasks. Proof of that is shown below.

Annotated screenshots demonstrating what you have achieved:

TASK 3.1:

Figure 1 showing screenshot showing the code used to complete task 3.1:

-The value in $3 is the operator, and the value in both $1 and $2 are the user inputted numbers. The program uses a series of 'if' and 'else-if' statements to check which operator has been typed in. The 'echo' statement prints the desired operation.

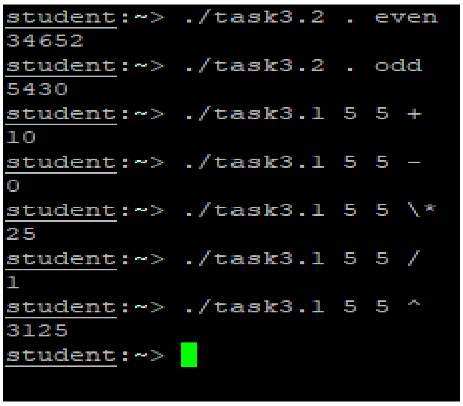


```
#!/bin/bash


if   [ "$3" == "+" ]
then
echo "$(($1 + $2))"

elif [ "$3" == "-" ]
then
echo "$(($1 - $2))"

elif [ "$3" == "*" ]
then
echo "$(($1 * $2))"

elif [ "$3" == "/" ]
then
echo "$(($1 / $2))"

elif [ "$3" == "^" ]
then
echo "$(($1 ** $2))"

fi
```

Figure 2 showing successful completion of task 3.1, which was to run the code to perform operations “+”, “-“, “*”, “/” “^”:

-The value in $1 is 5 and the value in $2 is 5. Expected results are as follows:

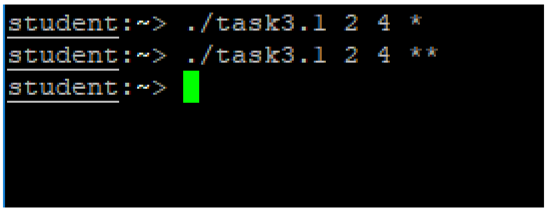
5+5 = 10
5-5 = 0
5x5 = 25
5/5 = 1
5^5 = 3125



```
student:~> ./task3.2 . even
34652
student:~> ./task3.2 . odd
5430
student:~> ./task3.1 5 5 +
10
student:~> ./task3.1 5 5 -
0
student:~> ./task3.1 5 5 \*
25
student:~> ./task3.1 5 5 /
1
student:~> ./task3.1 5 5 ^
3125
student:~>
```

Figure 3 showing possible errors that could be inputted:

-“*” and “**” are not operators so they will not display any answer



```
student:~> ./task3.1 2 4 *
student:~> ./task3.1 2 4 **
student:~>
```

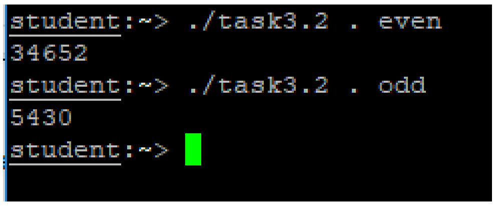
TASK 3.2:

Figure 4: completed code

We used a for loop for task 3.2 to open up the files and get the file sizes. Then, by using modulus to work out the remainder, we can find out if the file size is a odd or even number. This allows the script to process the file as odd or even. The line of code “((totalfilesize += $size))” adds the current file size to the total file size which is represented by the variable $size.

```
#!/bin/bash
totalfilesize=0
if [ "$2" == "odd" ]
then
           filesOdd="$1"/*
                for fOdd in $filesOdd;
                do
                       size=$(stat -c%s "$fOdd")

                        let "division=size%2"
                        if [ $division -eq 1 ]
                        then
                        ((totalfilesize += $size))
                        fi
                done
echo $totalfilesize


elif [ "$2" == "even" ]
then
                        filesEven="$1"/*
                for fEven in $filesEven;
                do


                       size=$(stat -c%s "$fEven")

                        let "division=size%2"
                        if [ $division -eq 0 ]
                        then
                        ((totalfilesize += $size))
                        fi
                done
echo $totalfilesize

fi
#test
```

Figure 5: Script running in putty



```
student:~> ./task3.2 . even
34652
student:~> ./task3.2 . odd
5430
student:~>
```

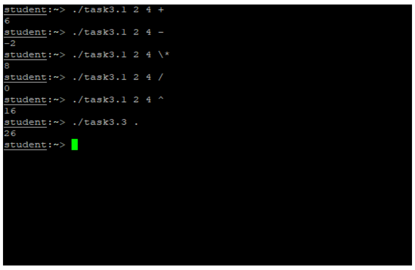
TASK 3.3:

Figure 6: completed code

In this script, we set the register to 0. Then by using the command `ls $1` we can open up the folder to display the files. By using a for loop, we can add in increments of 1 for each file present. “Echo $sum” will display the total number of files in a directory.

```
#!/bin/bash

sum=0

for file in `ls $1`
do
sum=$((sum +1))
done
echo "$sum"
```

Figure 7: Script running in putty:



```
student:~> ./task3.1 2 4 +
6
student:~> ./task3.1 2 4 -
-2
student:~> ./task3.1 2 4 \*
8
student:~> ./task3.1 2 4 /
0
student:~> ./task3.1 2 4 ^
16
student:~> ./task3.3 .
26
student:~> 
```

Personal Reflection:

This weeks' lab required us to use the Putty application once more to create and edit scripts just like last week. We were also more familiar with shell programming so the first task was not as difficult. We, however, ran into issues with the first task as we used the wrong operations in the Putty application, using "*" and "**" instead of "*" and "^" for multiplication and powers. These corrections and error tests are done and shown in figures 2 and 3. Regarding task 3.2, we ran into problems using script check initially after getting our code to run in Putty, as we did last week. After searching it up, we came to a conclusion that we were supposed fi for both task 3.1 and 3.2, as we were using if statements. We also added "#test" as a comment to make our code run in script check, as it was cutting off the last few lines of code when uploaded. Task 3.3 was fairly straight forward and did not give us any real problems that we hadn't previously encountered. I am glad that I experienced this lab session and 100% believe that is has made me a better programmer and given me a better understanding of system shells.

COMP1562 Logbook (Week 4)

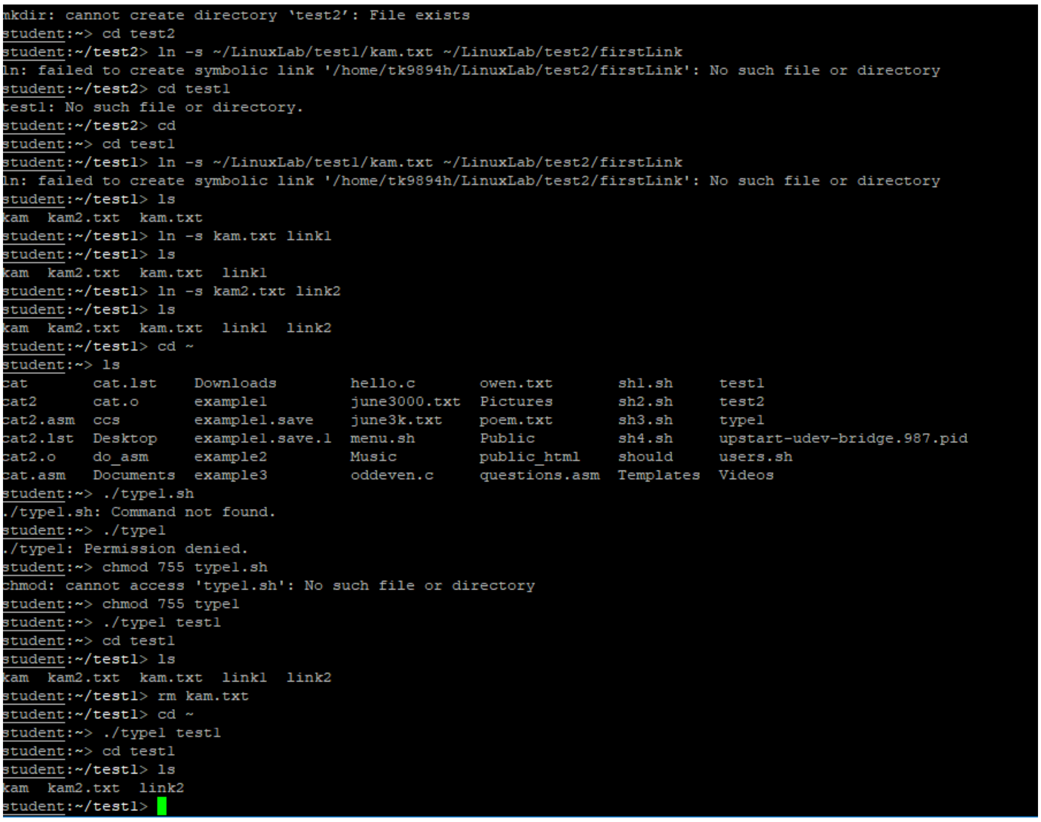
Basic Information

1.1 Student name	**001001720**
1.2 Who did you work with? Name and/or id	**001006731**
1.3 Which lab topic does this document relate to?	Process Handling
1.4 How well do you feel you have done?	I have completed the exercise to the best of my ability and I feel like I could've done better if given more time.
1.5 Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete most of the tasks. Proof of that is shown below.

Annotated screenshots demonstrating what you have achieved:

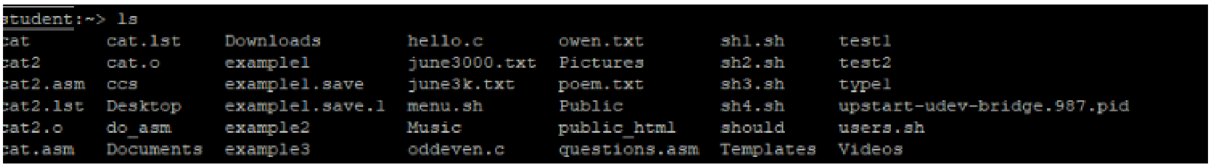
TASK 4.1:

Figure 1 showing screenshot showing the code used to complete task 4.1:
find . -xtype l -delete



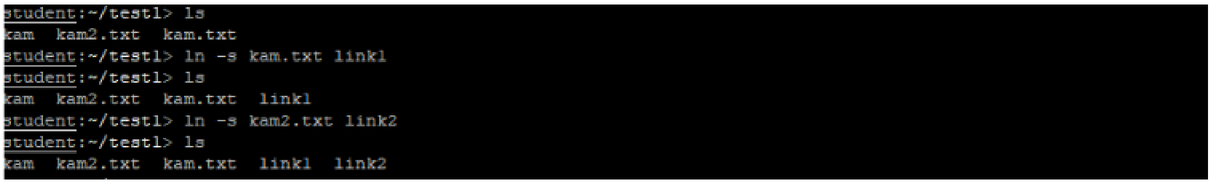
```
mkdir: cannot create directory 'test2': File exists
student:~> cd test2
student:~/test2> ln -s ~/LinuxLab/test1/kam.txt ~/LinuxLab/test2/firstLink
ln: failed to create symbolic link '/home/tk9894h/LinuxLab/test2/firstLink': No such file or directory
student:~/test2> cd test1
test1: No such file or directory.
student:~/test2> cd
student:~> cd test1
student:~/test1> ln -s ~/LinuxLab/test1/kam.txt ~/LinuxLab/test2/firstLink
ln: failed to create symbolic link '/home/tk9894h/LinuxLab/test2/firstLink': No such file or directory
student:~/test1> ls
kam  kam2.txt  kam.txt
student:~/test1> ln -s kam.txt link1
student:~/test1> ls
kam  kam2.txt  kam.txt  link1
student:~/test1> ln -s kam2.txt link2
student:~/test1> ls
kam  kam2.txt  kam.txt  link1  link2
student:~/test1> cd ~
student:~> ls
cat       cat.lst   Downloads        hello.c       owen.txt       sh1.sh     test1
cat2      cat.o     example1         june3000.txt  Pictures       sh2.sh     test2
cat2.asm  ccs       example1.save    june3k.txt    poem.txt       sh3.sh     type1
cat2.lst  Desktop   example1.save.1  menu.sh       Public         sh4.sh     upstart-udev-bridge.987.pid
cat2.o    do_asm    example2         Music         public_html    should     users.sh
cat.asm   Documents example3         oddeven.c     questions.asm  Templates  Videos
student:~> ./type1.sh
./type1.sh: Command not found.
student:~> ./type1
./type1: Permission denied.
student:~> chmod 755 type1.sh
chmod: cannot access 'type1.sh': No such file or directory
student:~> chmod 755 type1
student:~> ./type1 test1
student:~> cd test1
student:~/test1> ls
kam  kam2.txt  kam.txt  link1  link2
student:~/test1> rm kam.txt
student:~/test1> cd ~
student:~> ./type1 test1
student:~> cd test1
student:~/test1> ls
kam  kam2.txt  link2
student:~/test1>
```

The task was to delete broken symbolic linked (symlinked) files that were linked to previously linked deleted files. The commands we used were the “rm” command to remove the files, and ls to list the files.



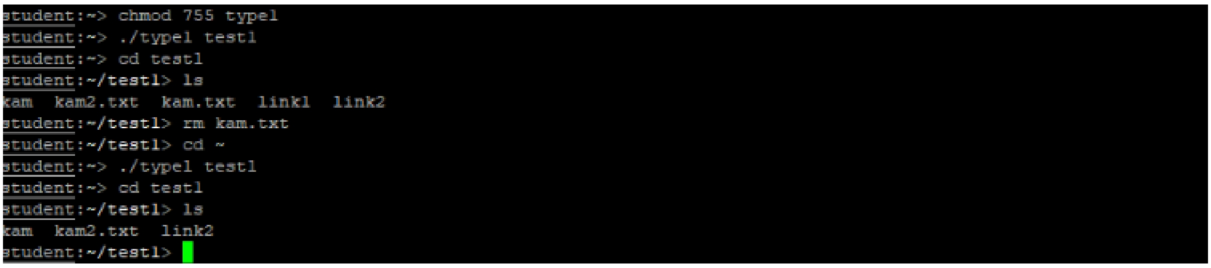
```
student:~> ls
cat        cat.lst    Downloads        hello.c        owen.txt       sh1.sh     test1
cat2       cat.o      example1         june3000.txt   Pictures       sh2.sh     test2
cat2.asm   ccs        example1.save    june3k.txt     poem.txt       sh3.sh     type1
cat2.lst   Desktop    example1.save.1  menu.sh        Public         sh4.sh     upstart-udev-bridge.987.pid
cat2.o     do_asm     example2         Music          public_html    should     users.sh
cat.asm    Documents  example3         oddeven.c      questions.asm  Templates  Videos
```

We created 2 files, named ‘kam.txt’ and ‘kam2.txt’, and put them in the first directory named ‘test1’. We then used the ‘ls’ command to list the files in the first directory to make sure they were there.



```
student:~/test1> ls
kam  kam2.txt  kam.txt
student:~/test1> ln -s kam.txt link1
student:~/test1> ls
kam  kam2.txt  kam.txt  link1
student:~/test1> ln -s kam2.txt link2
student:~/test1> ls
kam  kam2.txt  kam.txt  link1  link2
```

We then used the ‘rm’ command to get rid of the ‘kam .txt’ file and used ‘ls’ to list the files in the given directory again. Once listed, it can be seen that the ‘kam.txt’ file is no longer in the directory.



```
student:~> chmod 755 type1
student:~> ./type1 test1
student:~> cd test1
student:~/test1> ls
kam  kam2.txt  kam.txt  link1  link2
student:~/test1> rm kam.txt
student:~/test1> cd ~
student:~> ./type1 test1
student:~> cd test1
student:~/test1> ls
kam  kam2.txt  link2
student:~/test1> 
```

Personal Reflection:

This weeks’ task was a tricky one. Battled with the completion of the week 3 task after the extension, and the already released week 5 task, we found it really difficult to choose what to focus on. The lecture slides really helped with the completion of task 4.1, which like last week, was hard for us to implement in both scriptcheck and putty at the same time. However, after rethinking it with my team member, we were able to make it work. Much of our problem came from us choosing to do this task separately in effort to learn the material that would possibly show up in the future exam given to us in May. Next time, we will work together throughout the whole task to ensure it is completely finished like the previous weeks. Nevertheless, I still learned a lot about shell scripting and process handling, especially creating symbolic linked files. I am glad to have furthered my understanding of Operating Systems, and I look forward to fully completing the next task.

COMP1562 Logbook (Week 5)

Basic Information

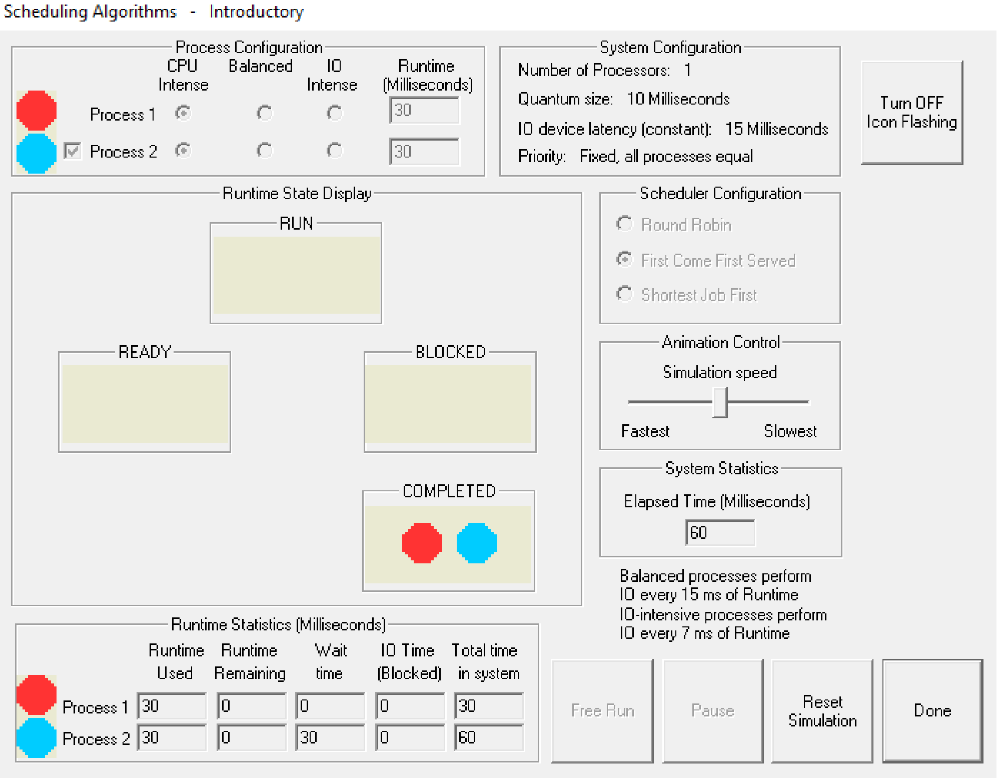
1.1 Student name	**001001720**
1.2 Who did you work with? Name and/or id	**001006731**
1.3 Which lab topic does this document relate to?	Scheduling
1.4 How well do you feel you have done?	I have completed the exercise and am totally satisfied with my work.
1.5 Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete the tasks. Proof of that is shown below.

Annotated screenshots demonstrating what you have achieved:

EXERCISES:

Exercises a-h required us to run the various algorithm simulations in Workbench in order to monitor the results. Screenshots of the various simulations are shown below.

a. FCFS1



Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	●	○	○	30
☑ Process 2	●	○	○	30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Scheduler Configuration

○ Round Robin

● First Come First Served

○ Shortest Job First

System Statistics

Elapsed Time (Milliseconds) 60

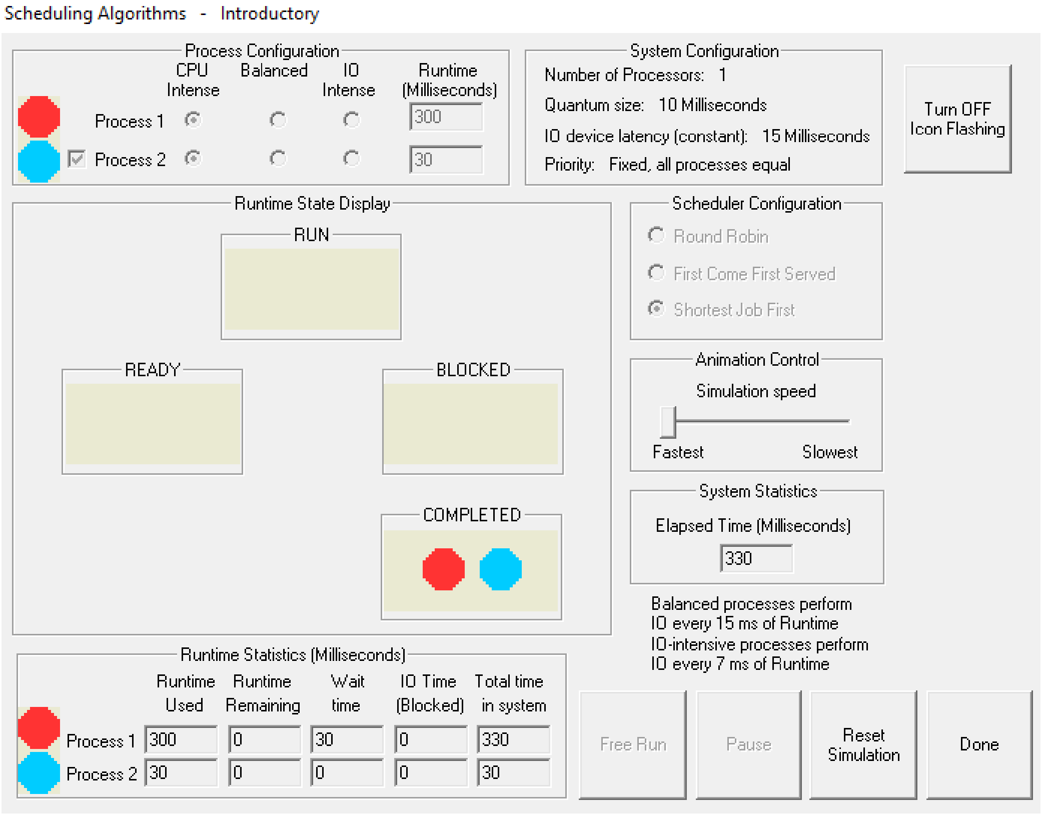
Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	0	30
Process 2	30	0	30	0	60

b. SJF 1



Scheduling Algorithms - Introductory

Process Configuration

		CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
🟥	Process 1	●	○	○	300
🟦	☑ Process 2	●	○	○	30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Scheduler Configuration

○ Round Robin

○ First Come First Served

● Shortest Job First

System Statistics

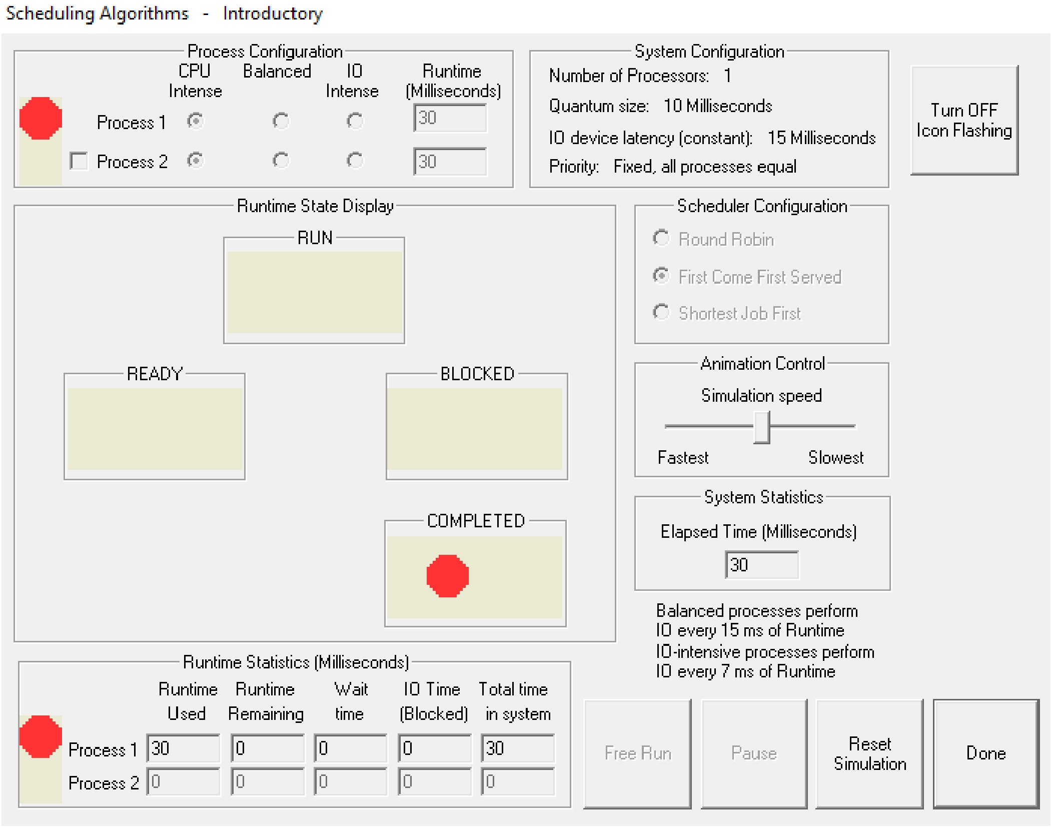
Elapsed Time (Milliseconds) 330

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

		Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
🟥	Process 1	300	0	30	0	330
🟦	Process 2	30	0	0	0	30

c. FCFS2



Scheduling Algorithms - Introductory

Process Configuration

		CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
🟥	Process 1	●	○	○	30
	□ Process 2	●	○	○	30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Scheduler Configuration

○ Round Robin

● First Come First Served

○ Shortest Job First

System Statistics

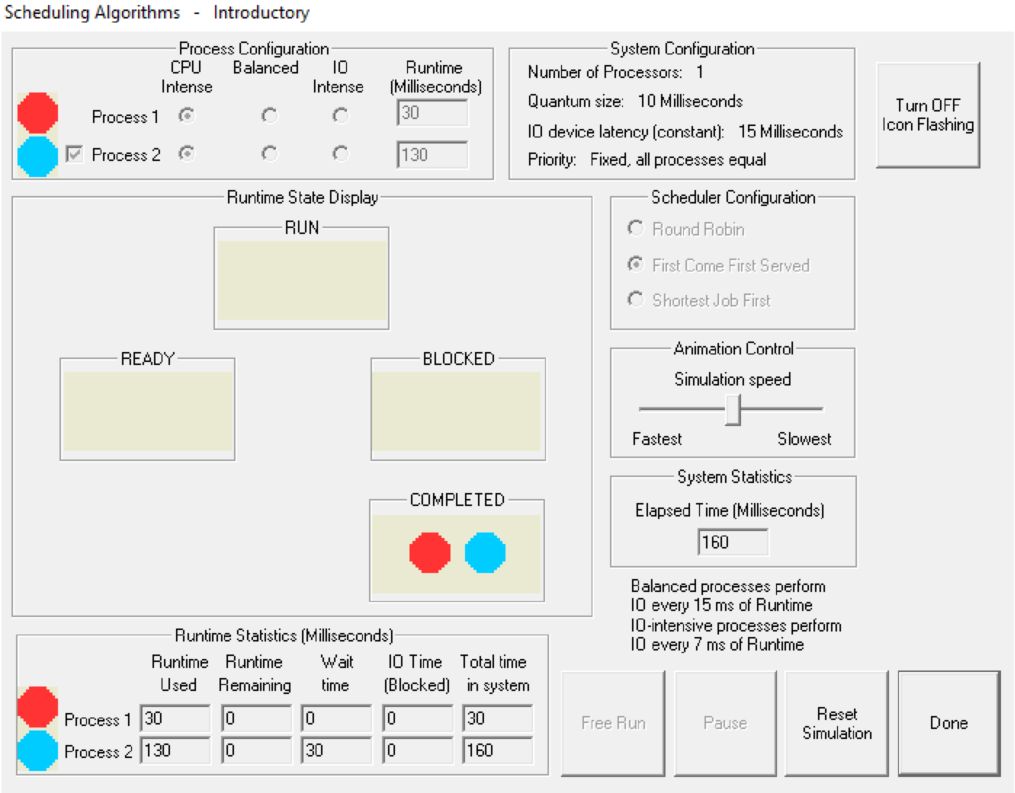
Elapsed Time (Milliseconds) 30

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

		Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
🟥	Process 1	30	0	0	0	30
	Process 2	0	0	0	0	0

d. FCFS3



Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	●	○	○	30
☑ Process 2	●	○	○	130

System Configuration
Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

Scheduler Configuration
○ Round Robin
● First Come First Served
○ Shortest Job First

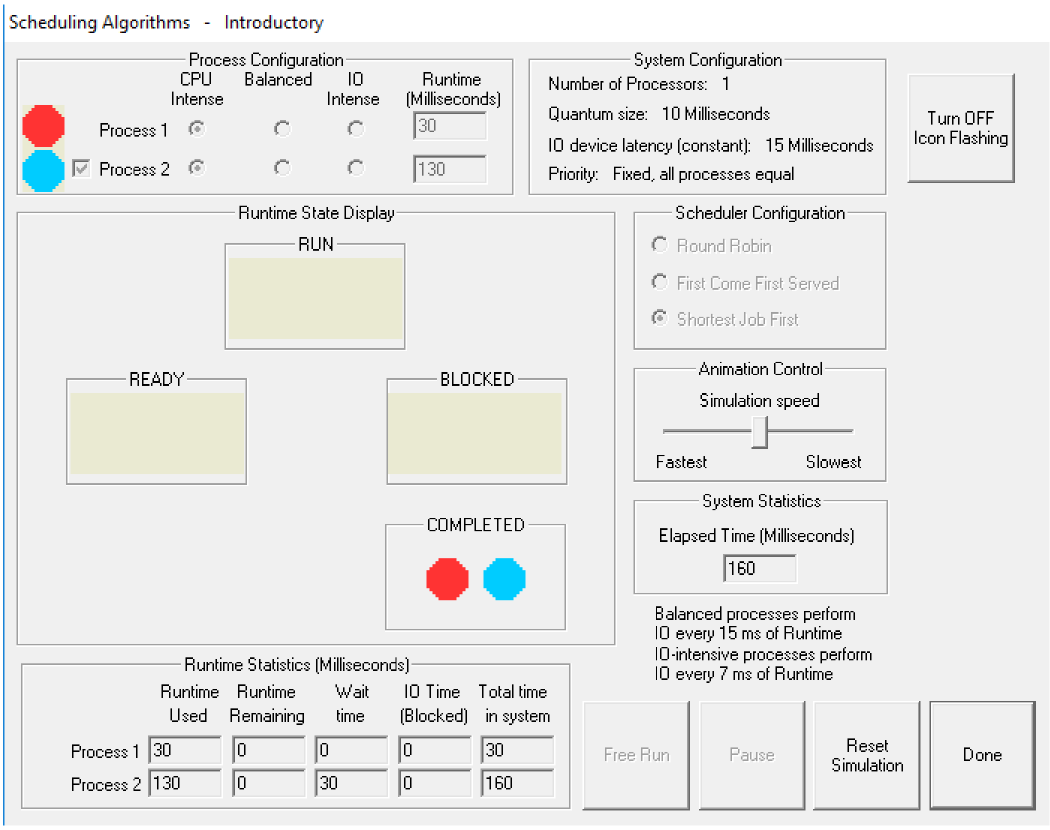
System Statistics
Elapsed Time (Milliseconds) 160

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	0	30
Process 2	130	0	30	0	160

e. SJF2



Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	●	○	○	30
☑ Process 2	●	○	○	130

System Configuration
Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

Scheduler Configuration
○ Round Robin
○ First Come First Served
● Shortest Job First

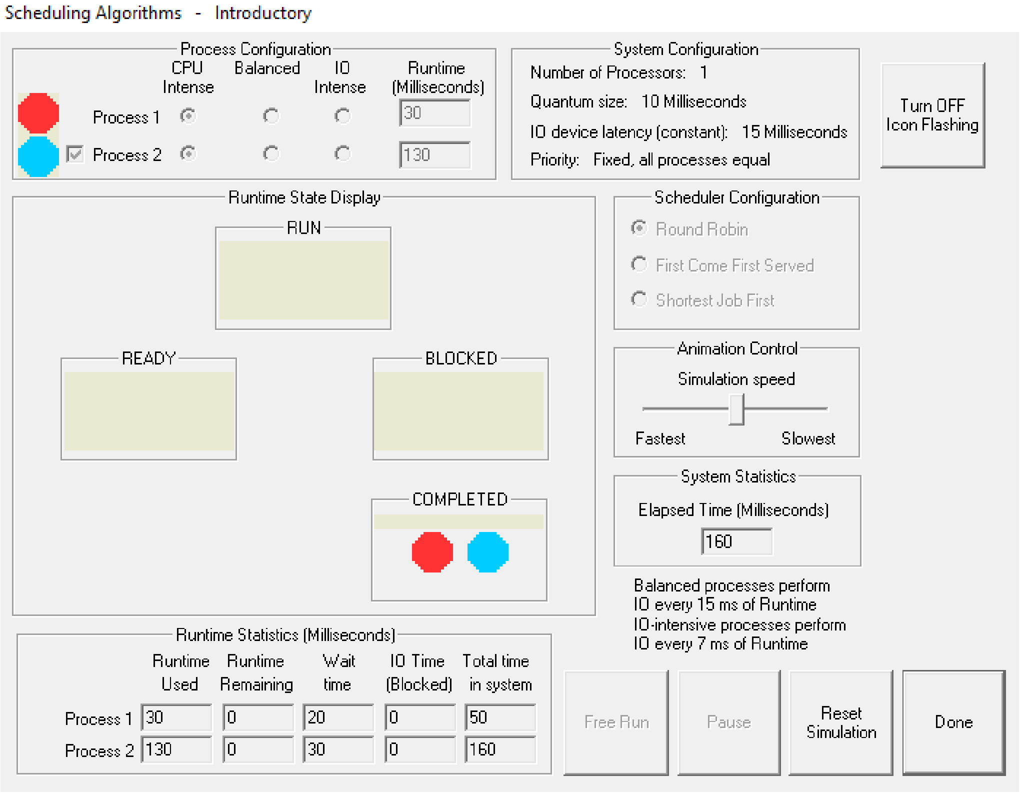
System Statistics
Elapsed Time (Milliseconds) 160

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	0	30
Process 2	130	0	30	0	160

f. RR1



Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	●	○	○	30
☑ Process 2	●	○	○	130

System Configuration
Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

Scheduler Configuration
● Round Robin
○ First Come First Served
○ Shortest Job First

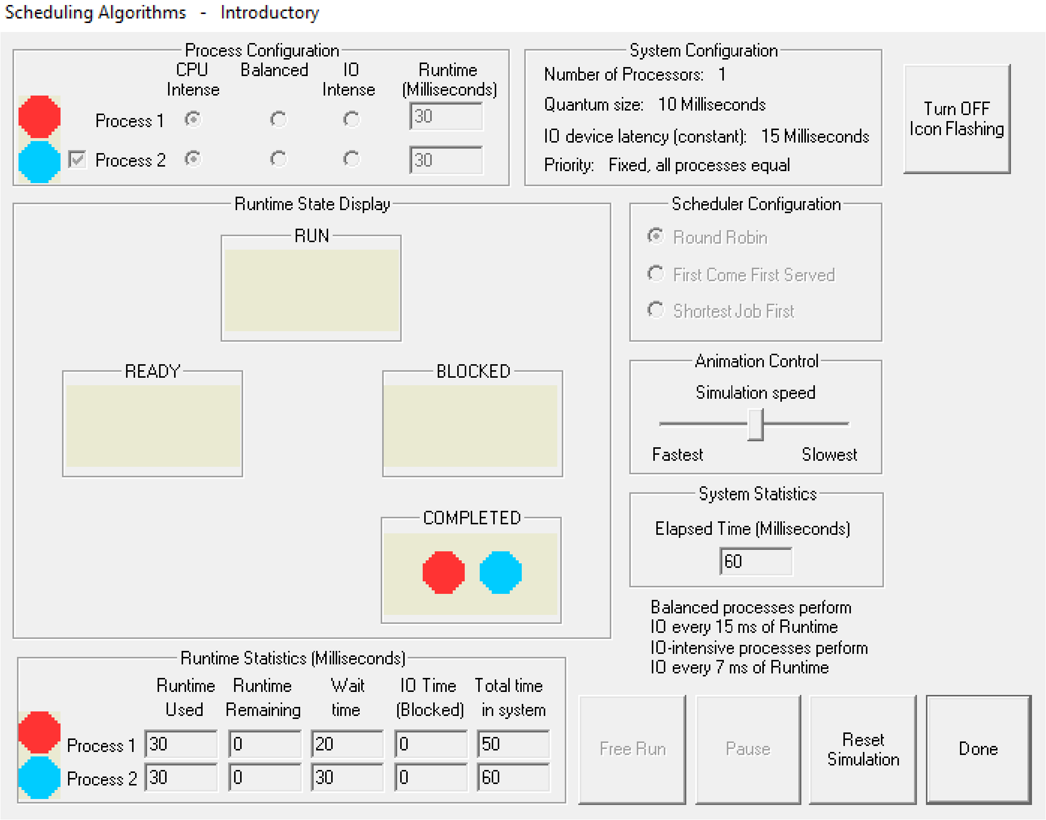
System Statistics
Elapsed Time (Milliseconds) 160

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	20	0	50
Process 2	130	0	30	0	160

g. RR2



Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	●	○	○	30
☑ Process 2	●	○	○	30

System Configuration
Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

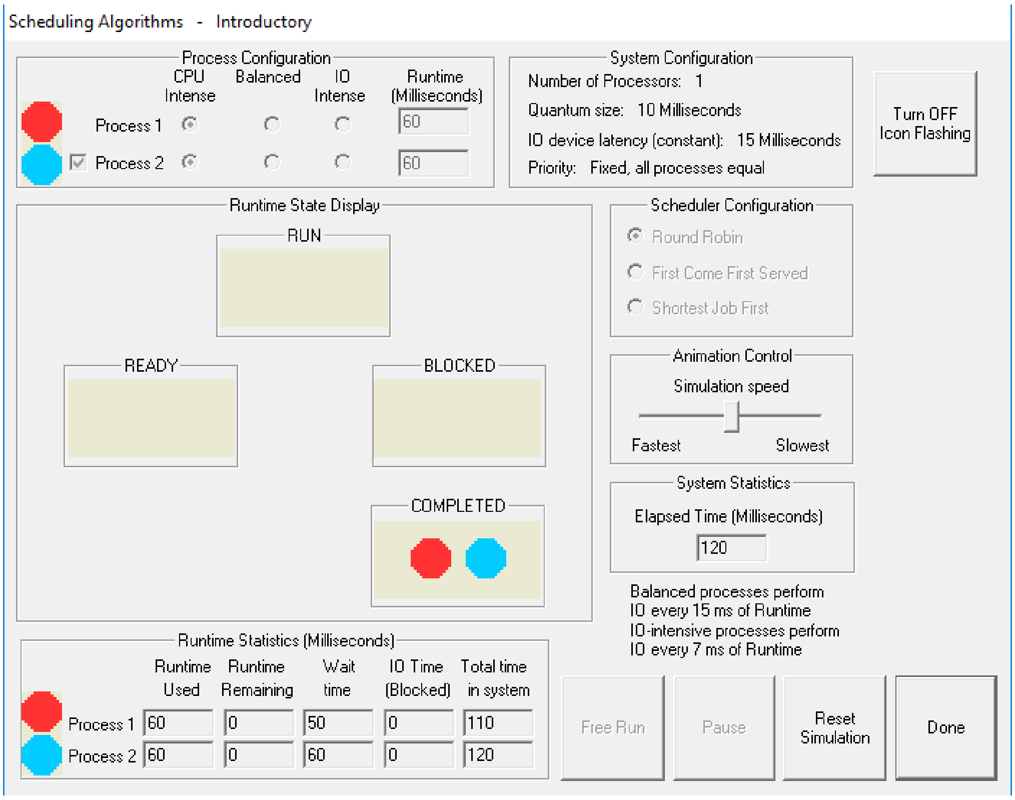
Scheduler Configuration
● Round Robin
○ First Come First Served
○ Shortest Job First

System Statistics
Elapsed Time (Milliseconds) 60

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	20	0	50
Process 2	30	0	30	0	60



Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	●	○	○	60
☑ Process 2	●	○	○	60

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Scheduler Configuration

● Round Robin

○ First Come First Served

○ Shortest Job First

System Statistics

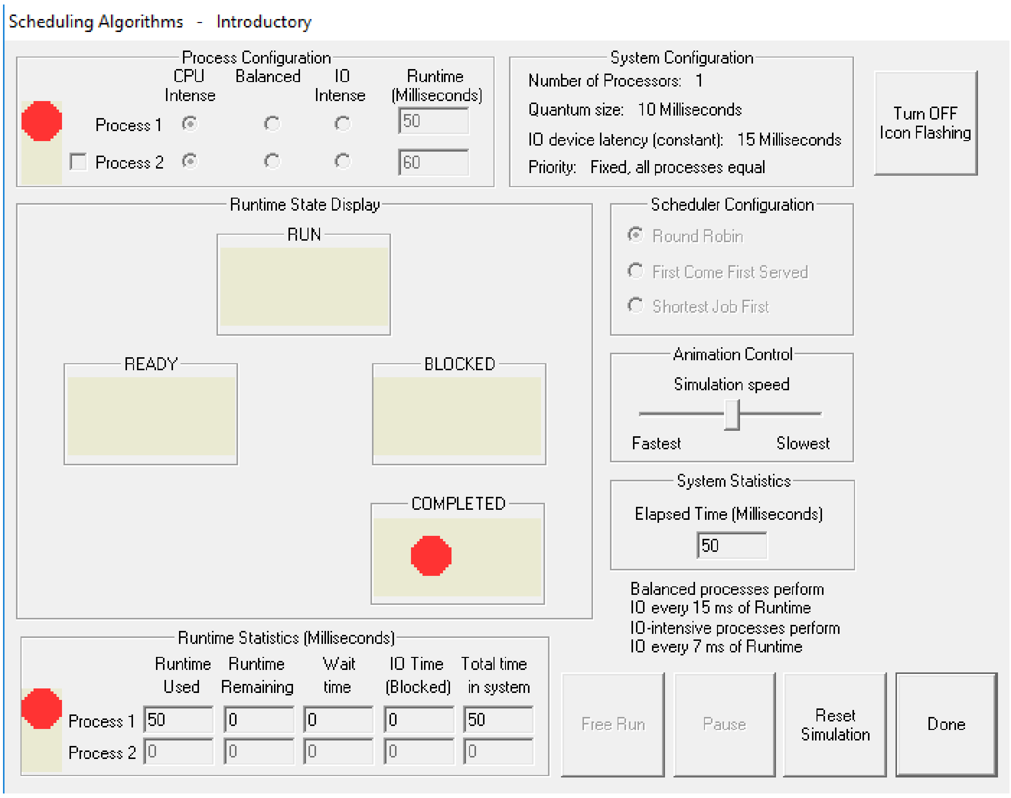
Elapsed Time (Milliseconds) 120

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	60	0	50	0	110
Process 2	60	0	60	0	120

h. RR3



Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	●	○	○	50
□ Process 2	●	○	○	60

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Scheduler Configuration

● Round Robin

○ First Come First Served

○ Shortest Job First

System Statistics

Elapsed Time (Milliseconds) 50

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	50	0	0	0	50
Process 2	0	0	0	0	0

TASKS:

For the following scenario with four processes please draw Gannt's charts and calculate *average waiting time* t_{AWT} and *average time a process remains in the system* t_{ATT} for the following scheduling algorithms:

SCENARIO:

Process	Arrival Time (When it arrives in the system) [ms]	Service Time (From arrival to finish) [ms]
P1	0	11
P2	2	8
P3	4	4
P4	6	6

TASK 5.1:

First Comes First Served:

The FCFS algorithm is shown below. The processes in the algorithm ran completely in order (P1, P2, P3, P4), with the total time in the system increasing as each process was carried out. The calculated average wait time for each process was 10.25 ms, and the average time a process remained in the system was 17.5 ms. The scenario table was unchanged, and the detailed t_{AWT} and t_{ATT} table, code, gantt chart and screenshots are shown below:

t_{AWT} and t_{ATT} table:

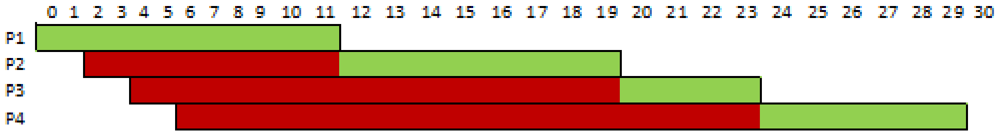
Process	Wait Time (t_{AWT}) [ms]	Average time a process remains in system (t_{ATT}) [ms]
P1	0	11
P2	9	17
P3	15	19
P4	17	23
Average:	10.25	17.5

Scriptcheck Code:

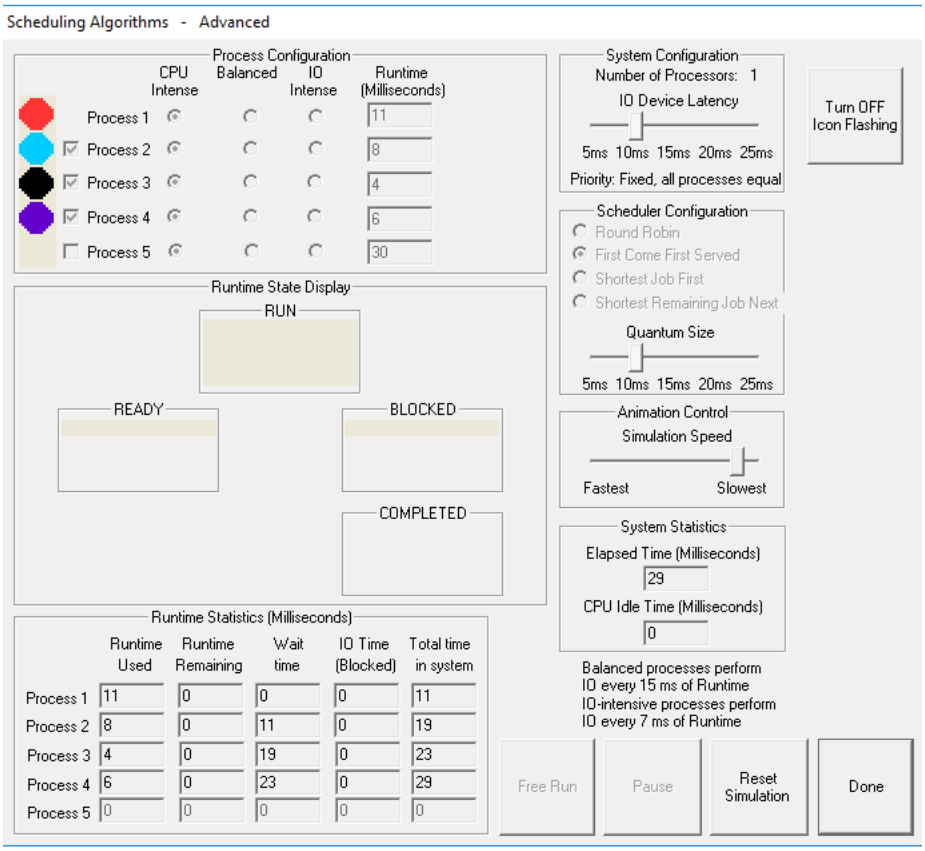
```
rrrrrrrrrrr
--wwwwwwwwwrrrrrrrr
----wwwwwwwwwwwwwwwrrrr
```

------wwwwwwwwwwwwwwwwwrrrrrr

Gantt Chart: (x-axis: process, y-axis: time[ms]):



Workbench Screenshot:



Scheduling Algorithms - Advanced

Process Configuration

		CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
	Process 1	●	○	○	11
☑	Process 2	●	○	○	8
☑	Process 3	●	○	○	4
☑	Process 4	●	○	○	6
□	Process 5	●	○	○	30

System Configuration
Number of Processors: 1
IO Device Latency
5ms 10ms 15ms 20ms 25ms
Priority: Fixed, all processes equal

Scheduler Configuration
○ Round Robin
● First Come First Served
○ Shortest Job First
○ Shortest Remaining Job Next
Quantum Size
5ms 10ms 15ms 20ms 25ms

Animation Control
Simulation Speed
Fastest Slowest

System Statistics
Elapsed Time (Milliseconds)
29
CPU Idle Time (Milliseconds)
0

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	11	0	0	0	11
Process 2	8	0	11	0	19
Process 3	4	0	19	0	23
Process 4	6	0	23	0	29
Process 5	0	0	0	0	0

TASK 5.2:

Shortest Job Next:

The SJN algorithm is shown below. The processes in the algorithm ran in a different order than FCFS (P1, P3, P4, P2), with the total time in the system fluctuating as each process was carried out. The calculated average wait time for each process was 8.75 ms, and the average time a process remained in the system was 16 ms. The updated scenario table, detailed t_{AWT} and t_{ATT} table, code, gantt chart and screenshots are shown below:

SCENARIO:

Process	Arrival Time (When it arrives in the system) [ms]	Service Time (From arrival to finish) [ms]
P1	0	11
P3	4	4
P4	6	6
P2	2	8

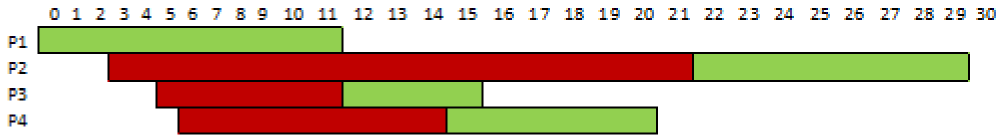
t_{AWT} and t_{ATT} table:

Process	Wait Time (t_{AWT}) [ms]	Average time a process remains in system (t_{ATT}) [ms]
P1	0	11
P2	9	15
P3	19	27
P4	7	11
Average:	8.75	16

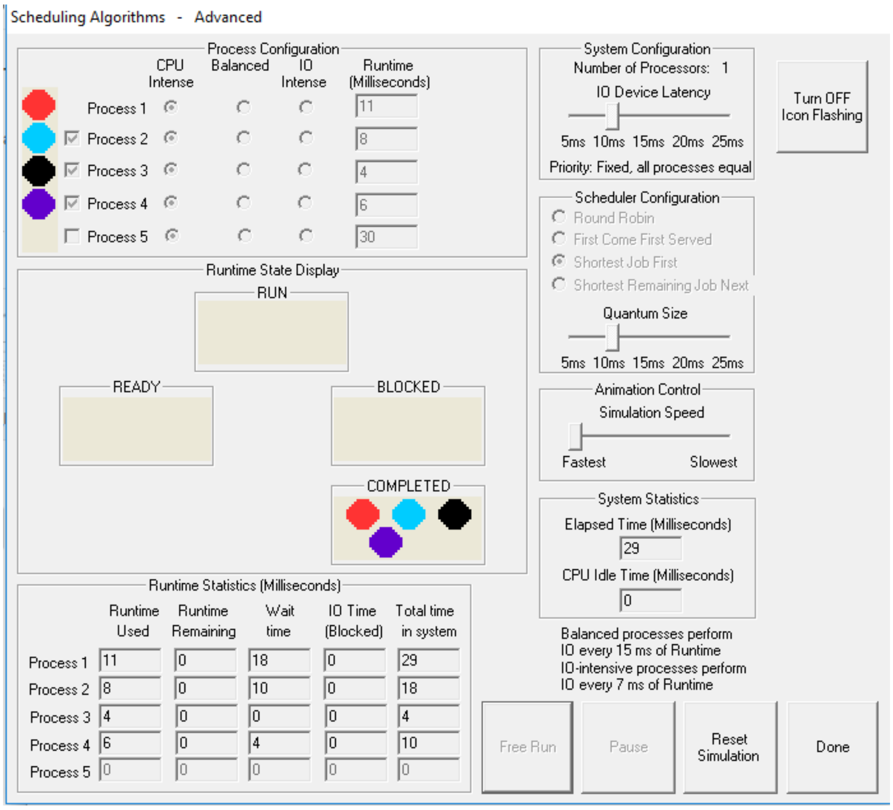
Scriptcheck Code:

```
rrrrrrrrrrr
----wwwwwwwrrrr
------wwwwwwwwwrrrrrr
--wwwwwwwwwwwwwwwwwwwrrrrrrrr
```

Gantt Chart: (x-axis: process, y-axis: time[ms]):



Workbench Screenshot:



Scheduling Algorithms - Advanced

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	◉	○	○	11
☑ Process 2	◉	○	○	8
☑ Process 3	◉	○	○	4
☑ Process 4	◉	○	○	6
□ Process 5	◉	○	○	30

System Configuration
Number of Processors: 1
IO Device Latency
5ms 10ms 15ms 20ms 25ms
Priority: Fixed, all processes equal

Scheduler Configuration
○ Round Robin
○ First Come First Served
◉ Shortest Job First
○ Shortest Remaining Job Next
Quantum Size
5ms 10ms 15ms 20ms 25ms

Animation Control
Simulation Speed
Fastest Slowest

System Statistics
Elapsed Time (Milliseconds) 29
CPU Idle Time (Milliseconds) 0

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	11	0	18	0	29
Process 2	8	0	10	0	18
Process 3	4	0	0	0	4
Process 4	6	0	4	0	10
Process 5	0	0	0	0	0

Personal Reflection:

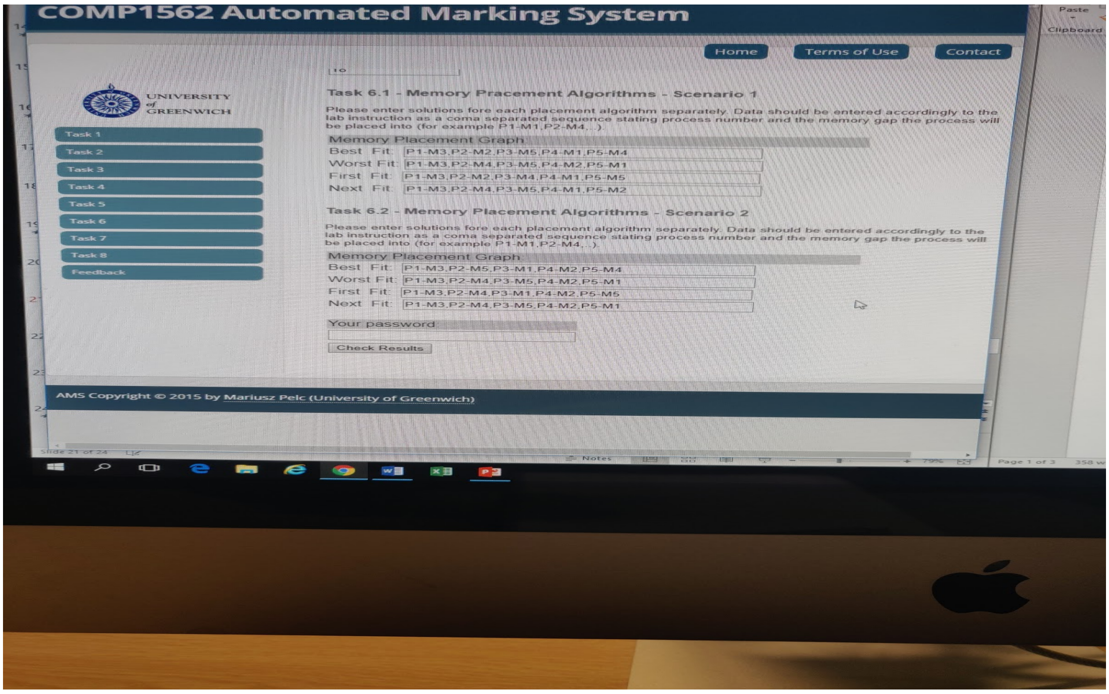
This weeks' task was surprisingly easy compared to last weeks'. We were required to familiarize ourselves with the WorkBench program, one which I had never encountered before. However, after running through the pre-task exercises, I was able to confidently use the program. I found it so cool that I could pause and simulate multiple processes and algorithms, mimicking how they would actually run in real time situations. Unlike the previous weeks' we had no problem using scriptcheck to upload out answers, as the input system was fairly straight forward. The tasks required us to learn what we had used in the exercises and apply it to different scenarios, which was easy enough. We had also made gantt charts in previous modules so that was an easy task as well. Calculating the wait and average time in the system was a matter of simple math, and before we knew it, we were finished with week 5 tasks. I thoroughly enjoyed the task and hope the future ones are as easy to understand as this one. I do believe that I have furthered my understanding of operating systems and look forward to continuing so next week.

COMP1562 Logbook (Week 6)

Basic Information

1.1 Student name	**001001720**
1.2 Who did you work with? Name and/or id	**001006731**
1.3 Which lab topic does this document relate to?	Memory placement algorithms
1.4 How well do you feel you have done?	I have completed the exercise and am totally satisfied with my work.
1.5 Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete the tasks. Proof of that is shown below.

Annotated screenshots demonstrating what you have achieved:



Task 6.1 - Memory Pracement Algorithms - Scenario 1

Please enter solutions fore each placement algorithm separately. Data should be entered accordingly to the lab instruction as a coma separated sequence stating process number and the memory gap the process will be placed into (for example P1-M1,P2-M4,...).

Memory Placement Graph:

Best Fit: P1-M3,P2-M2,P3-M5,P4-M1,P5-M4
Worst Fit: P1-M3,P2-M4,P3-M5,P4-M2,P5-M1
First Fit: P1-M3,P2-M2,P3-M4,P4-M1,P5-M5
Next Fit: P1-M3,P2-M4,P3-M5,P4-M1,P5-M2

Task 6.2 - Memory Placement Algorithms - Scenario 2

Please enter solutions fore each placement algorithm separately. Data should be entered accordingly to the lab instruction as a coma separated sequence stating process number and the memory gap the process will be placed into (for example P1-M1,P2-M4,...).

Memory Placement Graph:

Best Fit: P1-M3,P2-M5,P3-M1,P4-M2,P5-M4
Worst Fit: P1-M3,P2-M4,P3-M5,P4-M2,P5-M1
First Fit: P1-M3,P2-M4,P3-M1,P4-M2,P5-M5
Next Fit: P1-M3,P2-M4,P3-M5,P4-M2,P5-M1

Your password:

Check Results

AMS Copyright © 2015 by Mariusz Pelc (University of Greenwich)

TASK 6.1 and 6.2:

This week's task required us to simulate various memory allocation algorithms, those being first fit, best fit, next fit and worse fit.

First fit:

This method searched for the first available location. Though it executed quickly, it was not the most time efficient method.
The code is shown below:
P1-M3,P2-M2,P3-M4,P4-M1,P5-M5
P1-M3,P2-M4,P3-M1,P4-M2,P5-M5

Best fit:

This method searched for the best fitting process, which was good. This process, however, was very slow despite its efficiency, as it searched through all the blocks.
The code is shown below:
P1-M3,P2-M2,P3-M5,P4-M1,P5-M4
P1-M3,P2-M5,P3-M1,P4-M2,P5-M4

Next fit:

This method searched for the next available location that would fit the size of the specific process. It was very time efficient as it searched for the next location but not as memory efficient as it did not assess if there were better options for storage.
The code is shown below:
P1-M3,P2-M4,P3-M5,P4-M1,P5-M2
P1-M3,P2-M4,P3-M5,P4-M2,P5-M1

Worst fit:

This method searched for the most ill-fitting location. Though it executed quickly, it was the least efficient method, using the largest given memory block.
The code is shown below:
P1-M3,P2-M4,P3-M5,P4-M2,P5-M1
P1-M3,P2-M4,P3-M5,P4-M2,P5-M1

Personal Reflection:

This week's task was a relatively easy one as well. Just like the previous week, we were required to simulate algorithms to monitor the end result. I was confused about the task at first, but after consulting with my group member we were able to complete task 6.1. Task 6.2 was a bit tricky solely because of the fact that I did not realize that I had left some extra spaces in our scriptcheck solution. However, it was an easy fix and we were able to collect the full marks. I enjoyed the fact that the task made use of previous tasks, (using concepts such as memory registers, locations and process handling; these helped to efficiently introduce us to concepts such as memory allocation. I am happy with our performance this week and look forward to revising during skills week. I am confident that I have furthered my understanding of operating systems and look forward to learning more.

COMP1562 Logbook (Week 7)

Basic Information

1.1 Student name	**001001720**
1.2 Who did you work with? Name and/or id	**001006731**
1.3 Which lab topic does this document relate to?	File Systems
1.4 How well do you feel you have done?	I have completed the exercise and am totally satisfied with my work.
1.5 Briefly explain your answer to question 1.4	My group and I were able to successfully follow and complete the tasks. Proof of that is shown below.

Annotated screenshots demonstrating what you have achieved:

REPORT:

We were supposed to write a brief, coherent report (2-4 pages long) comparing, contrasting or describing the following disk monitoring and disk maintenance tools for Linux and Windows systems:

Linux and Windows Systems

fsck (Linux) vs **scandisk** (Windows)

fsck (Linux): fcsk (file system consistency check) is a tool that runs on unix/unix like systems like Linux and FreeBSD. In Linux however, it is mainly used a checking system, making sure that there are no errors in the file. It also attempts to repair fixable errors within the disk and generates a report of its findings. Fcsk is also used when a system fails to boot up and/or files or specific file location is compromised. Files are checked for inconsistencies that cause corrupted files.

scandisk (Windows): scandisk is a tool that runs on the Windows operating system. It is also used as an error checking system for softwares running the system. There are two main methods of error checking while using scandisk; error checking tool and SFC/SCANNOW. The error checking tool can be accessed using the File Explorer while SFC uses command lines to scan for files and resources.

Comparison between **fcsk** and **scandisk**: Both tools are quite similar, however scandisk can be limited as it only checks file systems supported in Windows. Fcsk is a bit more robust and much more optimised for deeper file checks.

mkfs (Linux) vs **format** (Windows)

mkfs (Linux): This is a command line line system used to build systems on the Linux operating systems. It is a directory based system, also creating sub-directories for the given storage device. The system does not create a directory hierarchy, allowing for the user to organize directories and files.

format (Windows): This is the preparation of a certain drive to be used by the Windows operating system (ex: formatting a hard drive). This is usually done using the disk management tool (often accessed by using a 'right-clicking' shortcut) or even in command prompt and control panel. This however, can only be done when the specified drive is not in use.

Comparison between **mkfs** and **format**: Once again, the Linux tool is a little bit more robust as it can use a variety of file systems, and can also uses up much less memory than Windows due to a process called 'Swap Partitioning'.

Logical Volume Manager (Linux) vs **Logical Disk Manager** (Windows).

Logical Volume Manager (Linux): This allows the user to view disk storage in greater detail, allowing the user to be able to make better decisions regarding storage use. It also allows the user to adjust 'storage volumes' that control multiple disk drives, making it as easy to do as if the user was only controlling smaller disk drives. This allows the user to also allocate larger groups of disk space to different drives as desired, giving the user an almost 100% customizable experience.

Logical Disk Manager (Windows): Similar to LVM in Linux, this allows the user to create and customize dynamic disks. These can be resized just like the 'storage volumes' in Linux; these are called dynamic volumes, of which there are 5 of:

● Simple - This dynamic volume is made from a single disk. A simple volume can be
made from a single region on a disk or multiple regions on the same disk that are
linked together.

● Mirrored - This dynamic volume is error free. If one of the disks fail, the data stored on the second disk and the system runs as usual.

● Spanned - This dynamic volume is made by using multiple physical disks. Size can be extended by using additional user created disks.

● Striped - This dynamic volume stores data in stripes on two or more physical drives.
Data storage on these devices is done in 'stripes'.

● RAID-5 - By far the most robust (except for Linux), it is error-free. 'Parity' allows errors and failures in the disks to be reconstructed. RAID-5 volumes can only be created on dynamic disks running Windows. It can also revive info from failed drives.

Comparison between **LVM** and **LDM**: The Windows based dynamic disks are a little more robust than the basic disks used in the LVM. Damaged disks can be replicated/reproduced without data loss, and the user can choose whether or not they would like to use the logical volume manager or the dynamic volume manager, allowing for more flexibility.

S.M.A.R.T. utility

SMART (Self-Monitoring, Analysis, and Reporting Technology) is a disk monitoring application used to scan hard drives to test operational and mechanical problems. The reason SMART utility is so coveted is because of its ability to display the individual attributes of the drive and specify exactly where it is failing. After this has been determined, it provides a report of where the drive failed and potential fixes if not already resolved. The fact that the system is self-testing means that some problems can be detected before they become too serious (ex: irreparable damage).

Comparison with others: SMART is very robust as it runs on modern and solid state disks. This allows the system to better detect/prevent problems with mechanical drives that are more prone to mechanical failures than software based drives.

References:

- Anon (2017) SMART Utility, *Volitans Software*, [online] Available at: https://www.volitans-software.com/apps/smart-utility/ (Accessed March 11, 2019).

- Anon (n.d.) NTFS: Logical Disk Manager documentation, *NTFS Transaction Journal*, [online] Available at: http://www.ntfs.com/ldm.htm (Accessed March 11, 2019).

- Anon (n.d.) *PH 213 Lab 3: Electric Potential [ph2xx]*, [online] Available at: http://physics.oregonstate.edu/~landaur/nacphy/coping-with-unix/node36.html (Accessed March 11, 2019).

- Both, D. (2016) A Linux user's guide to Logical Volume Management, *Opensource.com*, [online] Available at: https://opensource.com/business/16/9/linux-users-guide-lvm (Accessed March 11, 2019).

- Computer Hope (2018) How To Use Microsoft ScanDisk, *Computer Hope*, [online] Available at: https://www.computerhope.com/issues/ch001118.htm (Accessed March 11, 2019).

- Engel, van Kempen and Sommeling (n.d.) mkfs(8) - Linux man page, *pulse-daemon.conf(5) - Linux man page*, [online] Available at: https://linux.die.net/man/8/mkfs (Accessed March 11, 2019).

- Thomas (2011) Windows Spanned Disks (LDM) restoration with Linux?, *Stack Overflow*, [online] Available at: https://stackoverflow.com/questions/8427372/windows-spanned-disks-ldm-restoration-with-linux (Accessed March 11, 2019).

TASK 7.1:

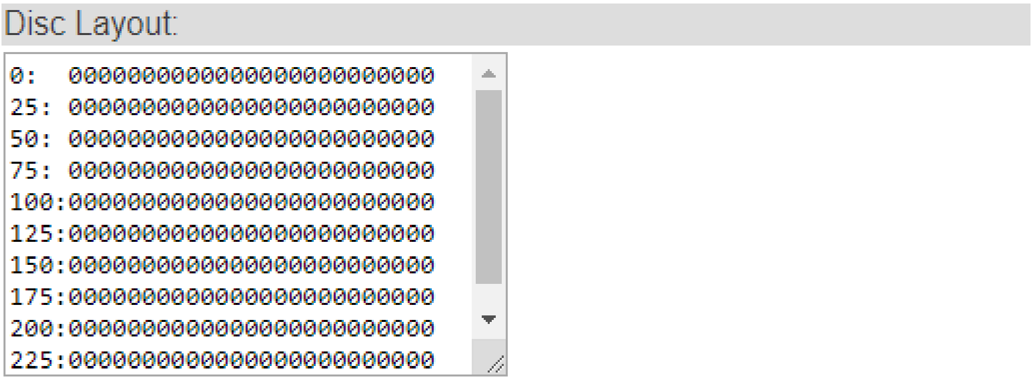
This task required us to allocate 3 files using the contiguous file allocation algorithm. We were also given 3 pre-allocated files that had already been allocated:

Pre-allocate 3 files are:
File X: size 15B, content: ‘%’, start block: 10
File Y: size 10B, content: ‘^’, start block: 30
File Z: size 33B, content: ‘&’, start block: 66

Number of files to allocate: 3
File A size: 14B, content: ‘a’,
File B size: 30B, content: ‘b’,
File C size: 12B, content: ‘c’.

Shown below is the default disk space template, pre-allocated disk space layout and the final layout that was uploaded to Scriptcheck:

Disk space template:
0:0000000000000000000000000
25:0000000000000000000000000
50:0000000000000000000000000
75:0000000000000000000000000
100:0000000000000000000000000
125:0000000000000000000000000
150:0000000000000000000000000
175:0000000000000000000000000
200:0000000000000000000000000
225:0000000000000000000000000



Disc Layout:

```
0:  0000000000000000000000000
25: 0000000000000000000000000
50: 0000000000000000000000000
75: 0000000000000000000000000
100:0000000000000000000000000
125:0000000000000000000000000
150:0000000000000000000000000
175:0000000000000000000000000
200:0000000000000000000000000
225:0000000000000000000000000
```

Pre-allocated disk space layout:

```
0:   00 00 00 00 00 %% %% %% %% %% %% %% %
25: 00 00 0^ ^^ ^^ ^^ ^^ ^0 00 00 00 00 0
50: 00 00 00 00 00 00 00 00 && && && && &
75:  && && && && && && && && && && && && 0
100: 00 00 00 00 00 00 00 00 00 00 00 00 0
125: 00 00 00 00 00 00 00 00 00 00 00 00 0
150: 00 00 00 00 00 00 00 00 00 00 00 00 0
175: 00 00 00 00 00 00 00 00 00 00 00 00 0
200: 00 00 00 00 00 00 00 00 00 00 00 00 0
225: 00 00 00 00 00 00 00 00 00 00 00 00 0
```

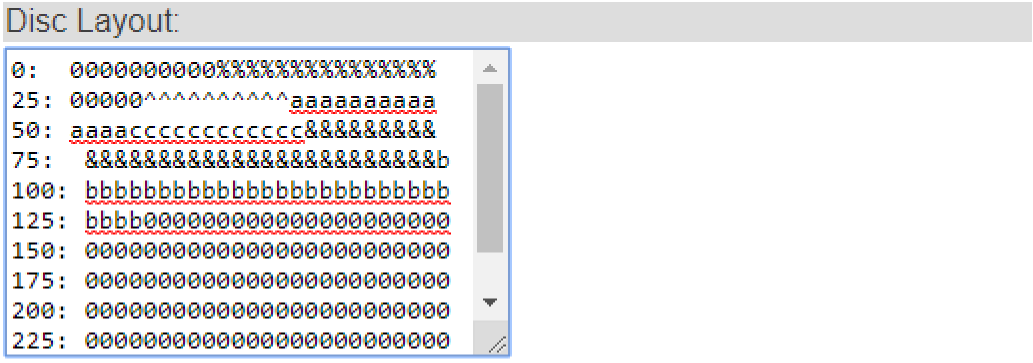
Final/Scriptcheck Layout:

```
0:  0000000000%%%%%%%%%%%%%%%
25: 00000^^^^^^^^^^aaaaaaaaaa
50: aaaacccccccccccc&&&&&&&&&
75:  &&&&&&&&&&&&&&&&&&&&&&&&b
100: bbbbbbbbbbbbbbbbbbbbbbbbb
125: bbbb000000000000000000000
150: 0000000000000000000000000
175: 0000000000000000000000000
200: 0000000000000000000000000
225: 0000000000000000000000000
```

Personal Reflection:

This task was on of the more enjoyable yet irritable ones, as the task at hand was not straight forward at first. We had two theories we it came to allocating the letters; My partner tried to replace the pre-allocated files with the new ones, while I tried to ‘integrate’ them with the new ones. The upload was a bit tricky because not matter what we tried, the last 3 lines of code would always be correct because the output was to be ‘0000000000000000000000000’. We then used the process of elimination and concluded that the way I was doing the task was correct, and we began integrating the files. A few checks on scriptcheck to confirm our answers and we were finished with the task. The report was pretty straight forward as we were given the terms to research. I am happy to have completed this task, and I feel like I have definitely furthered my understanding of Operating Systems because of it.

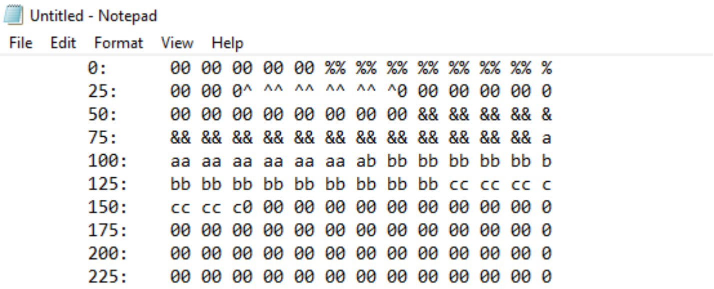
As usual, initial scriptcheck uploads were problematic as usual, but after we removed the the extra spaces between lines, the lines were able to function correctly. Editing the lines in notepad then transferring them helped:



Disc Layout:

```
0:   0000000000%%%%%%%%%%%%%%%
25:  00000^^^^^^^^^^aaaaaaaaaa
50:  aaaacccccccccccc&&&&&&&&&
75:   &&&&&&&&&&&&&&&&&&&&&&&&b
100:  bbbbbbbbbbbbbbbbbbbbbbbbb
125:  bbbb000000000000000000000
150:  0000000000000000000000000
175:  0000000000000000000000000
200:  0000000000000000000000000
225:  0000000000000000000000000
```

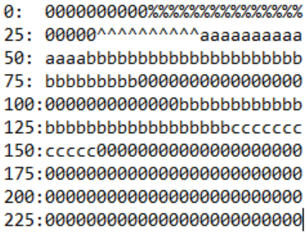
Typing the code in:



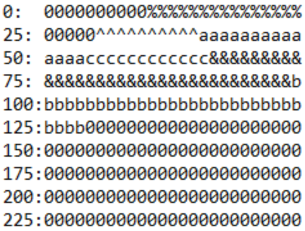
Untitled - Notepad

File Edit Format View Help

```
        0:          00 00 00 00 00 %% %% %% %% %% %% %% %
        25:         00 00 0^ ^^ ^^ ^^ ^^ ^0 00 00 00 00 0
        50:         00 00 00 00 00 00 00 00 && && && && &
        75:         && && && && && && && && && && && && a
        100:        aa aa aa aa aa aa ab bb bb bb bb bb b
        125:        bb bb bb bb bb bb bb bb bb cc cc cc c
        150:        cc cc c0 00 00 00 00 00 00 00 00 00 0
        175:        00 00 00 00 00 00 00 00 00 00 00 00 0
        200:        00 00 00 00 00 00 00 00 00 00 00 00 0
        225:        00 00 00 00 00 00 00 00 00 00 00 00 0
```



```
0:   0000000000%%%%%%%%%%%%%%%
25:  00000^^^^^^^^^^aaaaaaaaaa
50:  aaaabbbbbbbbbbbbbbbbbbbbb
75:  bbbbbbbbb0000000000000000
100:0000000000000bbbbbbbbbbbb
125:bbbbbbbbbbbbbbbbbbccccccc
150:ccccc00000000000000000000
175:0000000000000000000000000
200:0000000000000000000000000
225:0000000000000000000000000
```



```
0:   0000000000%%%%%%%%%%%%%%%
25:  00000^^^^^^^^^^aaaaaaaaaa
50:  aaaacccccccccccc&&&&&&&&&
75:  &&&&&&&&&&&&&&&&&&&&&&&&b
100:bbbbbbbbbbbbbbbbbbbbbbbbb
125:bbbb000000000000000000000
150:0000000000000000000000000
175:0000000000000000000000000
200:0000000000000000000000000
225:0000000000000000000000000
```