Name of Student			
Study Exp	7	**Roll No.**	
Date Of Perf.:		**Date Of Sub.:**	
Expt. Title	Explore OS Simulator		
CO Mapping			

Aim (1): At the end of this lab you should be able to:
1. Enter source code in the compiler and compile it to executable programs.

2. Load the programs into the CPU simulator's memory.

3. Create processes from programs in the OS simulator.

4. Select different scheduling policies and run the processes in the OS simulator.

5. Explain the differences between pre-emptive and non-pre-emptive scheduling.

Aim(2) : At the end of this lab you should be able to:
1. Construct a resource allocation graph for a deadlock condition and verify using the simulator.
2. Use two methods of resolving a deadlock condition.
3. Use two methods of preventing a deadlock condition.
4. Explain and use the "total ordering" method to prevent a deadlock.

Tutorial exercises 1

Learning outcome 1: Entering source code in the compiler and compiling it to an executable program.

You need to create some executable code so that it can be run by the CPU simulator under the control of the OS simulator. In order to create this code, you need to use the compiler which is part of the system simulator. This compiler is able to compile simple high-level source statements similar to Visual Basic. To do this, open the compiler window by selecting the **COMPILER...** button in the current window. You should now be looking at the compiler window.
In the compiler window, enter the following source code in the compiler's source editor window (under **PROGRAM SOURCE** frame title):

```
program LoopTest
i = 0
for n = 0 to 40
i = i + 1
next
end
```

Now you need to compile this in order to generate the executable code. To do this, click on the **COMPILE...** button. You should see the code created on the right in **PROGRAM CODE** view. Make a habit of saving your source code.

Click on the button **SHOW...** in **BINARY CODE** view. You should now see the **Binary Code for LOOPTEST** window. Study the program code displayed in hexadecimal format. Provide brief answers to the following questions and then close the window.

Learning outcome 2: Loading the program into the CPU simulator's memory.
Now, this code needs to be loaded in memory so that the CPU can execute it. To do this, first we need to specify a base address (in **ASSEMBLY CODE** view): uncheck the box next to the edit box with label **Base Address**, and then enter 100 in the edit box. Now, click on the **LOAD IN MEMORY...** button in the current window. You should now see the code loaded in memory ready to be executed. You are also back in the CPU simulator at this stage. This action is equivalent to loading the program code normally stored on a disc drive into RAM on the real computer systems.

Learning outcome 3: Create processes from programs in the OS simulator.
We are now going to use the OS simulator to run this code. To enter the OS simulator, click on the **OS 0...** button in the current window. The OS window opens. You should see an entry, titled **LoopTest**, in the **PROGRAM LIST** view. Now that this program is available to the OS simulator, we can create as many instances, i.e. processes, of it as we like. You do this by clicking on the **CREATE NEW PROCESS** button. Repeat this four times. Observe the four instances of the program being queued in the ready queue which is represented by the **READY PROCESSES** view. **Now, it is very important that you follow the instructions below without any deviation. If you do, then you must re-do the exercise from the beginning as any follow-up action(s) may give the wrong results**.

Learning outcome 4: Select different scheduling policies and run the processes in the OS simulator.

Learning outcome 5: Explain the differences between pre-emptive and non-pre-emptive scheduling.

1. Make sure the **First-Come-First-Served (FCFS)** option is selected in the **SCHEDULER/Policies** view. At this point the OS is inactive. To activate, first move the **Speed** slider to the fastest position, then click on the **START** button. This should start the OS simulator running the processes. Observe the instructions executing in the CPU simulator window.
processes are run (you need to concentrate on the two views: **RUNNING PROCESSES** and the **READY PROCESSES** during this period).

2. When all the processes finish, do the following. Select **Round Robin (RR)** option in the **SCHEDULER/Policies** view. Then select the **No priority** option in the **SCHEDULER/Policies/Priority** frame. Create three processes. Click on the **START** button and observe the behaviors of the processes until they all complete. You may wish to use speed slider to slow down the processes to better see what is happening. Make a note of what you observed and compare this with the observation in step 1 above.

3. Then select the **Non-preemptive** priority option in the **SCHEDULER/Policies/Priority** frame. Create three processes with the following priorities: 3, 2 and 4. Use the **Priority** drop down list to select priorities. Observe the order in which the three processes are queued in the ready queue represented by the **READY PROCESSES** view and make a note of this in the box below (note that the lower the number the higher the priority is).

4. Slide the **Speed** selector to the slowest position and then hit the **START** button. While the first process is being run do the following. Create a fourth process with priority 1. Make a note of what you observe (pay attention to the **READY PROCESSES** view).

5. Now kill all four processes one by one as they start running. Next, select the **Pre-emptive** option in the **SCHEDULER/Policies/Priority** frame. Create the same three processes as in step 3 and then hit the **START** button. While the first process is being run do the following. Create a fourth process with priority 1. Make a note of what you observe

Lab Exercises - Investigate and Explore

1. Four processes are running. They are called **P1** to **P4**.There are also four resources available (only one instance of each). They are named **R0** to **R3**. At some point of their existence each process allocates a different resource for use and holds it for itself forever. Later each of the processes request another one of the four resources. Draw the resource allocation graph for a four process deadlock condition.

2. In the compiler window, enter the following source code in the compiler source editor area (under **PROGRAM SOURCE** frame title).

```
program DeadlockP N
resource(X, allocate)
wait(3)
resource(Y, allocate)
for n = 1 to 20
next
end
```

The above code creates a program which attempts to allocate two resources for itself. After the first allocation it waits for 3 seconds and tries to allocate another resource. Finally it counts from 1 to 20 in a loop and then terminates. Other than that it does nothing sensible or useful!

Now follow the instructions below as faithfully as you can:

a. Copy the above code and paste it in three more edit windows so that you have a total of four pieces of source code.

b. In each case change **N** in the program name to 1 to 4, e.g. DeadlockP**1,** DeadlockP**2,** etc.

c. Look at your graph you constructed in (1) above and using that information fill in the values for each of the **X**s and **Y**s in the four pieces of source code.

d. Compile each one of the four source code.

e. Load in memory the four pieces of code generated.

f. Now switch to the OS simulator.

g. Create a single instance of each of the programs. You can do this by double-clicking on each of the program names in the **PROGRAM LIST** frame under the **Program Name** column.

h. In the **SCHEDULER** frame select **Round Robin (RR)** scheduling policy in the **Policies** tab.

i. In OS Control tab, push the speed slider up to the fastest speed.

j. Select the **Views** tab and click on the **VIEW RESOURCES...** button.

k. Select **Stay on top** check box in the displayed window.

l. Back in the **OS Control** tab use the **START** button to start the OS scheduler and observe the changing process states for few seconds.

m. Have you got a deadlock condition same as you constructed in (1) above? If you haven't then check and if necessary re-do above. Do not proceed to (n) or (3) below until you get a deadlock condition.
n. If you have a deadlock condition then click on the **SHOW DEADLOCKED PROCESSES...** button in the **System Resources** window. Does the highlighted resource allocation graph look like yours?

3. Now that you created a deadlock condition let us try two methods of getting out of this condition:
a. In the **System Resources** window, there should be four resource shapes that are in red colour indicating they are both allocated to one process and requested by another.
b. Select one of these resources and click on the **Release** button next to it.
c. Observe what is happening to the processes in the OS Simulator window.
d. Is the deadlock situation resolved? Explain briefly why this helped resolve the deadlock.
e. Re-create the same deadlock condition (steps in 2 above should help).
f. Once the deadlock condition is obtained again do the following: In the OS Simulator window, select a process in the waiting queue in the **WAITING PROCESSES** frame.
g. Click on the REMOVE button and observe the processes.
h. Has this managed to resolve the deadlock? Explain briefly why this helped resolve the deadlock.

This part of the exercises was about two methods of **recovering** from a deadlock condition **after** it happens.

4. We now look at two methods of **preventing** a deadlock condition **before** it happens.
a. In the **System Resources** window select the **Disallow hold and wait** check box in the **Prevent** frame.
b. Try to re-create the same deadlock condition as before. Have you been successful? What happened? Click on the **SHOW DEADLOCKED PROCESSES...** button and observe the displayed information in the text window for potential clues.
c. Next, uncheck the **Disallow hold and wait** check box and check the **Disallow circular wait** check box.
d. Try to re-create the same deadlock condition as before. Have you been successful? What happened? Click on the **SHOW DEADLOCKED PROCESSES...** button and observe the displayed information in the text window for potential clues.

5. We are now going to try a third method of preventing deadlocking before it happens. It is called "total ordering" method. Here the resources are allocated in increasing resource id numbers only. So, for example, resource R3 must be allocated after resources R0 to R2 and resource R1 cannot be allocated after resource R2 is allocated. Looking at your resource allocation graph can you see how this ordering can prevent a deadlock?
a. In the **System Resources** window select the **Use total ordering** check box in the **Prevent** frame. The other options should be unchecked.
b. Try to re-create the same deadlock condition as before. Have you been successful? What happened? Click on the **SHOW DEADLOCKED PROCESSES...** button and observe the displayed information in the text window for potential clues. What happened?

Evaluation:

Timeline (2)	Understanding(4)	Performance (4)	Total(10)

Date & Signature of teacher: **Students Signature:**