OPERATING SYSTEMS
COURSEWORK HANDOUT

UNIT DETAILS			
DEPARTMENT	Computer Science	PROGRAMME	BSc (Hons) Accredited by BCS (British Computer Society)
UNIT CODE	CCP2220	UNIT TITLE	Operating Systems
SEMESTER	Spring	SESSION	Spring 2019
STAFF	Dr K. Dimopoulos	STAFF OFFICE	6th floor L.Sofou building
E-MAIL	k.dimopoulos@citycollege.sheffield.eu		

COURSEWORK DETAILS					
C/W NUMBER	1	CONTRIBUTION	30% of the unit final mark		
C/W TITLE	Simulation of a simple OS				
C/W TYPE	Individual				
HAND-OUT DATE	26/4/19 Week 8	SUBMISSION DATE	25/5/19 Week 11	FEEDBACK DATE	8/6/19 Exam period

LEARNING OUTCOMES

Upon completion of this piece of coursework, a student will be able to:

- Describe the basic concepts of operating systems
- Be familiar with the theory behind the design of basic concepts
- Comprehend key issues in the practical implementation of operating systems
- Realise the concepts of multi-tasking and time-sharing
- Perform systems programming (processes) related tasks
- Solve OS design problems
- Be able to evaluate and select appropriate algorithms and data structures
- Enhance students programming skills

ASSESSMENT CRITERIA

The assessment of the project will be on the submission of the deliverables given in section “ASSESSMENT”. These deliverables will be used to assess:

- The analysis of the different algorithms discussed
- The implementation of these algorithms
- The experimentations and observations of how the workings of a simple OS are affected by these algorithms
- The quality of the report

If you have any questions about the course work, you may either contact me after my lectures or email me at the address provided at the top.

DETAILED DESCRIPTION

The aim of this coursework is to implement a number of process schedulers and memory managers, and to experiment with different combinations in order to see how they interact. To this end a **Simple Operating System** simulator (SOS) is provided to you, and it is made of the following components:

- The kernel (core of the operating system),
- a memory manager
- a process scheduler, and
- a process table where information about the running processes is kept.

The simulator is provided as a java library and can be used in a Processing project to run. A template project is provided to you as a .zip file at the course website. The provided Simulator can run only **First Fit Memory Manager** with **First Come First Serve process scheduler**. Running the simulator as it is, you will be able to see these two algorithms working together.

Your task is to **write the java classes that implement additional process schedulers and memory managers using the interfaces provided**. Once these are completed you will be able to experiment with many combinations. I suggest you read the comments in the code I provide to you thoroughly. You do not need to worry about the code in the classes not provided to you. **It is highly recommended that you write your own testers that test the methods you create.**

You are therefore required to:

1. Download the processing project and run it to see it working
2. Develop any of the algorithm combinations listed below (choose one from a to c):
    a. The **Next Fit** memory manager (including coalescing and compacting) AND the **Round Robin** process scheduler.
    b. The **Next Fit** memory manager (including coalescing and compacting) AND the **Shortest Remaining Time Next** process scheduler.
    c. The **Best Fit** AND the **Worst Fit** memory managers (including coalescing and compacting) AND the **Shortest Job First,** process scheduler.
3. Experiment with different combinations of process schedulers and memory managers, gather statistics and make useful observations. The simulator allows for parameterization of memory size, list of process as input and round robin slice size.
4. Write a small report describing your understanding of the selected algorithms (including those I provide to you in the simulator), your implementation and your experiments. TOTAL SIZE OF REPORT 2000 words.

The simulator includes a build in set of processes that you may use in all your simulations, but also supports the reading of a text file with the processes.

The simulator:
You only need to edit the setup() in the SOS.pde file. If you run the project as it is the simulator will run with only the First Come First Serve and the First Fit options. In comments I give you the code examples for changing the memory size manually to whatever size you want, and to add your implementations of the algorithms you chose. Make sure that the Memory manager is passed the same size of memory as you defined when you define the RAM size.

Your implementation:
The process scheduler needs to implement the IProcessScheduler.java interface. Read the method descriptions for what each method must do. Similarly the Memory manager needs to implement the IMemoryManager.java interface. All java files you implement need to be at the same level as the SOS.pde file. You may need to implement additional .java files at your liking.

The experiments:
You will need to be creative here and think what it is you want to observe. Some suggestions are: How do these algorithms cope with different size of RAM? If you have round robin what happens as the slices increase? Remember I give you the FCFS and First Fit algorithms to compare with.

The Report:
Finally each student will write an individual report with the following structure:

- Part 1: explain how the algorithms you chose to implement work in theory, giving examples. What assumptions do you take (for example what happens when a process is blocked),
- Part 2: how did you implement the algorithms, what choices you took that affected your implementation, data structures, methods etc. Include a class diagram, as well as **code snippets** (with comments), **with explanations reflecting your work**.
- Part 3: Describe the experiments you did with proper justification of what you hoped to observe by doing this comparison. Describe you observations and try to explain them. Was what you observed what you expected? What insights do your observations offer?
- The rest of the report should be organized as any formal report (include a short introduction, conclusions and proper **academic references**).

ASSESSMENT

Code:

- Implementation of the selected process scheduler [15%]
- Implementation of the selected memory manager(s) excluding compacting and coalescing [15%]
- Implementation of compacting and coalescing [10%]

Report:

- Theoretical description of the selected algorithms with detailed examples [10%]
- Explanation of your code and design choices [15%]

- Choice of experiments and justification [10%]
- Observations on the experiments performed [15%]
- Report structure and use of proper references [10%]

SUBMISSION

You are required to submit:

1. A report to Turnit in (see details bellow)
2. Your complete project folder as a .zip file uploaded to MOLE (in assessment part)

By not later than the date specified above.

FURTHER DETAILS

It is your responsibility to get this work done on time. Rules for late submission of individual deliverables do not apply. Individual deliverables are delivered in person to the lecturer on the day defined at the “ASSESSED DELIVERABLES” section, using the provided template (see website). For group deliverables rules for late submission apply as outlined in the 1st Level Handbook of the Computer Science Dept. That is, any practical (in this case deliverable) delayed will be marked with a penalty of 5% of the total mark per day of delay, up to 7 calendar days. Any practical submitted with a delay of more than 7 calendar days will be given a mark of zero. Note that submission date is the date and time written by the secretary on your practical submission form.

TURN IT IN REQUIREMENT

This piece of coursework is required to be submitted to **turnitin** plagiarism detection software at **www.turnitin.com** at a date no later than the submission date. This is an absolute requirement for releasing a mark. Brief instructions on how you can set up your profile and submit your work can be found at:

http://www.turnitin.com/static/training_support/tii_student_qs.pdf (text)
http://www.turnitin.com/static/support_video_tutorials.html (video)

You are going to require:

CLASS ID:	21077405	**ENROLLMENT PASSWORD:**	ccp2200

If you have any problems in submitting your work, please contact the course administrator or the lecturer.

NOTE

All sentences or passages quoted in this coursework from other people's work should be specifically acknowledged by clear cross-referencing to author, work and page(s). Failure to do this amounts to plagiarism and will be considered grounds for failure in this coursework. It is on the instructor’s discretion to contact an oral examination, which will result to the award of the final grade to that particular piece of coursework.