Introduction

This assignment should give you some idea how the Ada scheduler works, especially the occurrence of task switches, and the role of delay statements. The implementation of the protected stack of last week’s assignment is required.

Exercise 1:

Write a main program with 3 tasks, similar to the test program last week. The tasks are all identical, and receive a unique identifier at initialisation. The tasks should do the following:

1) Push 10 objects onto the stack. The objects should contain the task’s identifier.
2) Pop 10 objects from the stack, and print them to the screen. The text output on the screen should contain the task’s id, and the object taken from the stack.
3) Terminate.

Find out what happens in the following cases:

a) with no delay statements at all
b) with one delay statement between phase 1 (push) and phase 2 (pop)
c) with additional delay statements inside the loops (after each push/pop operation).

Is there a difference in the results between using `delay 0;` statements, `delay` statements with a fixed time, and `delay` statements with different times for different tasks?

What happens if the stack size is only 15 elements? In which cases could there be problems?

Exercise 2:

Add a non-blocking `Try Push` and a non-blocking `Try Pop` operation to your protected stack. They should return a boolean variable, which is true, if the operation was successful, and false otherwise (i.e. the stack is full, or empty).

Use these operations to implement the following test program. Again, there are three identical tasks with a unique identifier. Here, the tasks should do the following:

1) Push as many objects onto the stack as possible, using the non-blocking operation (until it is full). The objects should contain the task’s identifier.
2) Pop as many objects as possible from the stack, using the non-blocking operation, and print them to the screen. The text output on the screen should contain the task’s id, and the object taken from the stack.
3) Terminate.

Again, analyse the cases outlined in exercise 1.