Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330 - Final exam
Real-Time Applications and Embedded Systems - $1^{st}$ semester - 2023/24 - duration: 150 minutes

**Instructor:** Dr. Hanna Bullata

Student Name/ID: _____

## Q1: Bottle-Spinning Game simulation (40 points)

We would like to build a multi-processing application that simulates a bottle-spinning game that involves a group of 8 players. The 8 equidistant players are seated as a circle on the ground with a bottle in the middle that will be spinned by one of the players.

The simulation behaves as follows:

**A.** Initially a random player will spin the bottle in the middle with random spinning speed.

**B.** Once the bottle stops spinning, we check if the bottle's head is pointing towards a particular player. If not (meaning that the bottle is pointing between any 2 players), go to step **A** above. Else, the player at the bottle's head will ask the player on the opposite side of the bottle (at the bottle's tail) to do one of the following:

1. Do a quick dance.
2. Do a funny sketch or drawing.
3. Sing a small song.
4. Tell a joke.
5. Do a stupid thing.
6. Re-shuffle the players so as to change how the players are seated.

In every round, each player is allowed to do one of the above first 5 list actions only once. In addition, each player is allowed to do action **6** only twice per game.

**C.** During a round, a player who has done the first 5 list actions is called the round loser and that player's score is decremented by 5. In addition, the other players' scores are incremented in an inversely proportional way according to the number of actions they did during the round (e.g. a player's score is incremented by 5 if he did none of the above first 5 list items, or incremented by 4 if he did 1 list action, etc).

**D.** When a round is over, the round loser is allowed to spin the bottle in order to start a new round.

**E.** Go to step **B** and continue playing.

**J.** The simulation ends if any of the following is true:

- One of the players has lost 3 times in a row (3 losses in sequence).
- One of the players has a score above +31. We'll call that player the game winner.
- One of the players has a score below -31. We'll call that player the game loser.
- The players have made rounds more than a user-defined threshold.
- The simulation has been running for more than one hour.

## What you should do

- Create a folder named `soft` and put all what comes next under that folder.
- We would like to implement the above problem on Linux machines using a **multi-processing approach**.
- In few bullet table items, describe in human language how you would implement the above described problem (20 steps max). Name the file `description.txt`. If you use chatgpt or a similar tool to generate the description, you'll get a zero! **(15 points)**

- Using C-language, write the code for the bottle-spinning game. No need to compile. Just focus on the code logic and make sure all included items behave as expected. **(25 points)**

## Q2: PICMicro controller (30 points)

We would like to build a simple hardware system that is composed of the following components:

- PIC16F877A.
- $16 \times 2$ LCD.
- A push button $P$.

The behavior of the system can be explained as follows:

1. On start-up, the LCD screen displays on the first row the phrase `Enter Name`. The cursor is next positioned at the first column in the second row.
2. The character `A` is displayed at the cursor location.
3. With every click of push button $P$, the character is incremented by 1 (meaning character `A` becomes `B`, then `C`, etc). Of course when character `Z` is reached, an extra click of $P$ will display character `A` again.
4. When $P$ is left unclicked for more than 2 seconds, the cursor on the second LCD row shifts to the column to the right with respect to the current position.
5. When 8 characters are displayed on the second LCD row or when $P$ is double-clicked, the character input phase is over.
6. Now when $P$ is clicked, the above-entered message is displayed on the second row in reverse order (e.g. `MUSTAFA` is displayed as `AFATSUM`). Clicking $P$ again reverses the order over and over again.
7. When $P$ is double-clicked, go to step **1** above.

## What you should do

- Create a folder named `pic` and put all what comes next under that folder.
- In a file called `procedure.txt`, explain **in a tabular format** how you would connect the above-described system.
- Build a schematic using Proteus for the above-described system. Once done, make a **print-screen and save it as an image called** `schematic`.
- Using MPLAB application, create a project and name the main C-file `proj.c`. Assume you're using a 4MHZ oscillator. Insert comments to add clarity and make sure you place all project files under the folder you created above

## Grading

| **Task** | **Max grade** | **Grade** |
|---|---|---|
| The procedure sounds reasonable | **10** | |
| The LCD connections to PICMicro are correct | **3** | |
| The push button $P$ is well connected | **2** | |
| The code is well structured and has fair comments | **12** | |
| The application runs as expected | **3** | |
| **Total** | **30** | |

## Q3: FreeRTOS: Task coordination (30 points)

We would like to simulate the behavior of a production machine for which tasks have to be executed according to the following scenario using the multitasking approach and IPC techniques you've seen in FreeRTOS:

- When a push button $P$ is turned ON, the below activities take place. Otherwise, the machine is halted.
- `Task-1`, `Task-2` and `Task-3` have to execute first according to the following scenario:
  - `Task-1` has to execute first. It pulls pin 1 high for 2 seconds and then pulls it low.
  - Once `Task-1` has pulled pin 1 high, `Task-2` and `Task-3` will execute in any order. `Task-2` and `Task-3` will pull pin 2 and pin 3 high as long as `Task-1` keeps pin 1 high. When `Task-1` pulls pin 1 low, `Task-2` and `Task-3` both must pull their pins low. We say that both tasks have finished when both pins 2 & 3 are driven low.
- Once the first 3 tasks have finished, `Task-4` can run by pulling pin 4 high. It will keep it high as long as tasks `Task-5`, `Task-6` & `Task-7` have transitioned all their pins from low to high and then back to low. The delay between transitions is half-a-second.
- The above scenario repeats forever every 2 minutes as long as push button $P$ is not turned OFF. If at any point during the machine operation $P$ is turned OFF, halt all operations. Upon turning ON the machine later on, the tasks resume from the beginning.

## What you should do

- Create a folder named `freertos` and put all what comes next under that folder.
- Build the system described above using FreeRTOS approach.
- Use FreeRTOS IPC technique(s) to help build the application.
- Use notepad++ editor to write your code. On the top of the code file, describe in tabular format the steps that you would do in human language before you start your code.

## Grading

| **Task** | **Max grade** | **Grade** |
|---|---|---|
| Solution description steps | **10** | |
| FreeRTOS tasks | **5** | |
| FreeRTOS IPC technique(s) | **5** | |
| The code is correct & well structured | **10** | |
| **Total** | **30** | |

When exam time is over, send the zipped file that contains the folders `soft`, `pic` and `freertos` as a reply to my message entitled **encs4330 final exam - 1stSem 2023/24**.