*Respostas da SUPOSTA prova de HOJE kkkk:

Elabore, em linguagem C, duas funções, uma para envio sincronizado e temporizado de mensagens e outra para recepção temporizada de mensagens. A comunicação é armazenada e deve acontecer através de canal de comunicação. A mensagem deve ser do tipo inteiro.

```
int send_sync_timed (int *buffer =, int c, float tempo){
	time_t inicio, fim;
	canal[c] = *buffer;
	inicio = time(null);
	do{
		fim = time(null);
	} while((canal[c] = -1) && (difftime(fim, inicio) <= tempo)
	if(canal[c] == -1)
		return 0;
	else
		return 1;
}

int receive_timed (int *buffer =, int c, float tempo){
	time_t inicio, fim;
	inicio = time(null);
	do{
		fim = time(null);
	} while((canal[c] == -1) && (difftime(fim, inicio) <= tempo)
	if(canal[c] != -1){
		*buffer = canal[c];
		canal[c] = -1;
		return 0;
	}else
		return 1;
}
```

Elabore, em linguagem C, uma função que gere atraso ou retardo, cujo parâmetro é o tempo, em segundos, do atraso.

```
time.h
void delay(float tempo){
	time_t inicio, fim;
	inicio = time(null);
	do {
		fim = time(null);
	} while (difftime (fim, inicio) <= tempo)
		return;
}
```

Explique, com detalhes, o que faz o escopo de código em linguagem C para PIC abaixo. Qual o tempo, em segundos, do timeout?

```
char getc_timeout();
{
    long int tempo;
    erro_timeout = false;
    while(!kbhit() && (++tempo<500) delay_ms(10);
    if (kbhit()) return (getc());
    else{
        erro_timeout = true;
        return (0);
    }
}
```

R = - Verifica se há algum dado sendo recebido pela USART (serial padrão). Tempo de 5 segundos

- Ele aguarda alguma tecla ser pressionada durante um certo tempo, ele aguarda uma tecla e retorna a tecla. ele aguarda a resposta do khbit (kbhit era de keyboard hit kkkkkkk) faz isso, durante 5 segundos, depois ele verifica se houve resposta, caso nao, da timeout caso sim retorna getc();
- A correspondente função de entrada é getc (o nome é uma abreviatura de get character). Cada chamada da função lê um byte do arquivo especificado. (Muitas vezes, o byte lido representa um caractere ASCII.) Por exemplo, getc (stdin) lê o próximo byte do teclado e devolve o byte lido.

Mais questões:

Explique, com detalhes, o que faz o escopo de código em ADA abaixo:

```
task CONTROL is
   entry CALL(V:VOLTAGE)
end CONTROL;
task body CONTROL is
begin
  loop
     select
        accept CALL(V:VOLTAGE) do
            NEW_VOLT:=V;
         end CALL;
      or
         delay 10.0;
      end select;
   end loop;
end CONTROL;
```

R = O escopo de códico acima é a implementação de uma rotina de timeout, uma restrição no tempo de um processo que está esperando por um evento. A utilização de timeout é muito importante para detecção em falhas na passagem de mensagens entre processos, na eliminação de deadlocks, entre outros.

Elabore, em linguagem C, duas funções, uma para alocação de recursos, e outra para liberação de recursos. As funções podem ter os seguintes nomes e argumentos:

Allocate (int size); Free (int size). Observação: deve ser utilizado semáforo para sincronização de acesso aos recursos. A aplicação pode ter vários recursos disponíveis.

2. Elabore, em linguagem C, duas funções para operação sobre semáforos binários: uma função wait(semáforo) e uma função signal(semáforo):

```
Wait (int semaforo) {
  while(semáforo ==0);
  semaforo = semaforo – 1;
}

Signal(semaforo) {
  semaforo = semaforo + 1;
}
```

isso utilizando busy wait loop

4. Implemente, em linguagem C, uma função de envio sincronizado e uma função de recepção por canal. Apresente também um processo origem concorrente enviando a mensagem “50” para um processo destino concorrente. As funções devem ter essa sintaxe: send_sinc(int *msg, int índice) e receive(int * msg, int índice). Inicialize o canal:

```
canal = [-1, -1, -1];

void send_sinc(int *msg, int índice) {
  canal[índice] = msg;
  while(canal[índice] != -1) {};
}

void receive(int * msg, int índice) {
  while(canal[índice] == -1) {};
  msg = canal[índice];
  canal[índice] = -1;
}

P1() {
  send_sinc(50, 2);
}

P2() {
  int buffer;
```

```
  receive(buffer, 2);
}
```