Problema C

Escalonamento por EDF
Versão 1.0

Problem

Neste problema será fornecido com os dados sobre um conjunto de processos periódicos de tipo tempo real, nomeadamente o nome do processo, seu tempo de execução/computação (burst-time) em termos de unidades de CPU de cada processo e o período de execução

Agora o objectivo é escrever um programa que simula a execução do sistema para um sistema operativo de tempo real usando o algoritmo de escalonamento preemptivo "EDF" (Earliest Deadline First). Em cada instante do tempo deverá indicar em qual o processo a correr e a sua instância (Isto é chamado o "trace"). cada vez que um processo é instanciado dever ter associado um PID (Processo ID) que é um interior que começa com zero e incremente por unidades.

No fim dizer se houve "deadline miss" ou não.

Input

Uma linha com um inteiro N indicando o numero de processos.
A seguir, N linhas com nome dum processo (um carácter apenas) seguido por dois interios , Tempo de Execução (T) e (P) Período (P).
Todos os processo começam no tempo zero.
Finalmente uma linha com um inteiro S indicando o tempo de simulação. Exemplo

```
//number of processes
3
//process details
A 1 4
B 2 6
C 3 8
//Time for Simulation Run
25
```

Limites: Nota que
T,P e N são inteiros de 1-10 e S é um inteiro de 1-200

Note pode ler uma linha com scanf("%c%d%d")...

Output

O output é em primeiro lugar um linha com o valor da carga,um valor real de três pontos decimais "CARGA=%4.23" Segue depois um "trace" da simulação mostrando no inicio de cada unidade de tempo qual o processo e sua instância que está a executar. O trace terminará no tempo S Cada linha do "trace" tem o formato

```
T=IDLE
ou
T=VALOR PID=VALOR CPU=nome-instância
Algo assim "T=%d PID=%d CPU=%s\n"
```

No fim de simulação deverá imprimir uma mensagem "FALHA" ou "COMPTENTE"

Detecção da falha do prazo. Este tem que ser feito de seguinte maneira - quando um processo termine e apenas nesta situação um processo pode saber assim se falhou o seu prazo temporal ou não.

Samples

Sample Input 1

```
2
A 2 5
B 5 8
20
```

Sample Output 1

```
CARGA=1.025
T=0 PID=0 CPU=A-1
T=1 PID=0 CPU=A-1
T=2 PID=1 CPU=B-1
T=3 PID=1 CPU=B-1
T=4 PID=1 CPU=B-1
T=5 PID=1 CPU=B-1
T=6 PID=1 CPU=B-1
T=7 PID=2 CPU=A-2
T=8 PID=2 CPU=A-2
T=9 PID=3 CPU=B-2
T=10 PID=4 CPU=A-3
T=11 PID=4 CPU=A-3
T=12 PID=3 CPU=B-2
T=13 PID=3 CPU=B-2
T=14 PID=3 CPU=B-2
T=15 PID=3 CPU=B-2
T=16 PID=5 CPU=A-4
T=17 PID=5 CPU=A-4
T=18 PID=6 CPU=B-3
T=19 PID=6 CPU=B-3
COMPETENTE
```

O mesmo exemplo com S=50 irá dar uma FALHA (detectado no A-8)

Sample Input 2

```
2
A 2 5
B 4 8
15
```

Sample Output 2

```
CARGA=0.900
T=0 PID=0 CPU=A-1
T=1 PID=0 CPU=A-1
T=2 PID=1 CPU=B-1
T=3 PID=1 CPU=B-1
T=4 PID=1 CPU=B-1
T=5 PID=1 CPU=B-1
T=6 PID=2 CPU=A-2
T=7 PID=2 CPU=A-2
T=8 PID=3 CPU=B-2
```

```
T=9 PID=3 CPU=B-2
T=10 PID=4 CPU=A-3
T=11 PID=4 CPU=A-3
T=12 PID=3 CPU=B-2
T=13 PID=3 CPU=B-2
T=14 IDLE
COMPETENTE
```

Sample Input 3

```
3
A 1 4
B 2 6
C 3 8
25
```

Sample Output 3

```
CARGA=0.958
T=0 PID=0 CPU=A-1
T=1 PID=1 CPU=B-1
T=2 PID=1 CPU=B-1
T=3 PID=2 CPU=C-1
T=4 PID=2 CPU=C-1
T=5 PID=2 CPU=C-1
T=6 PID=3 CPU=A-2
T=7 PID=4 CPU=B-2
T=8 PID=4 CPU=B-2
T=9 PID=5 CPU=A-3
T=10 PID=6 CPU=C-2
T=11 PID=6 CPU=C-2
T=12 PID=6 CPU=C-2
T=13 PID=7 CPU=A-4
T=14 PID=8 CPU=B-3
T=15 PID=8 CPU=B-3
T=16 PID=9 CPU=A-5
T=17 PID=10 CPU=C-3
T=18 PID=10 CPU=C-3
T=19 PID=10 CPU=C-3
T=20 PID=11 CPU=B-4
T=21 PID=11 CPU=B-4
T=22 PID=12 CPU=A-6
T=23 IDLE
T=24 PID=13 CPU=A-7
COMPETENTE
```

Sample Input 4

```
3
A 6 12
B 7 16
C 4 10
20
```

Sample Output 4

```
CARGA=1.337
T=0 PID=2 CPU=C-1
T=1 PID=2 CPU=C-1
T=2 PID=2 CPU=C-1
T=3 PID=2 CPU=C-1
T=4 PID=0 CPU=A-1
T=5 PID=0 CPU=A-1
T=6 PID=0 CPU=A-1
T=7 PID=0 CPU=A-1
T=8 PID=0 CPU=A-1
T=9 PID=0 CPU=A-1
T=10 PID=1 CPU=B-1
T=11 PID=1 CPU=B-1
T=12 PID=1 CPU=B-1
T=13 PID=1 CPU=B-1
T=14 PID=1 CPU=B-1
T=15 PID=1 CPU=B-1
T=16 PID=1 CPU=B-1
T=17 PID=3 CPU=C-2
T=18 PID=3 CPU=C-2
T=19 PID=3 CPU=C-2
FALHA
```

Sugestão

Uma boa técnica de resolução é de similar mesmo o funcionamento dum sistema operativo de tempo real !
Define uma estrutura de dados para representar os processos e os seus estados, um *Processo Controlo Bloco* com campos para

- Numero de Processo (Process ID = PID)
- Nome do processo e numero de instancia
- Tempo de Chegada/Inicio
- Tempo de Prazo (Deadline)
- tempo de duração inicial / Burst Time Inicial
- tempo de CPU restante (ainda para executar)
- .. etc

Definir uma estrutura de dados para os processos que estão ativas, prontos a executar - processos no estado "ready" e mão se esqueça do processo que está a executar. Depois faça um ciclo de unidade de tempo fazendo as decisões apropriadas no inicio de cada unidade de tempo e no fim da unidade de tempo.

Entrega e Programação em Grupos

Pode realizar este programa e o problema a seguir D em grupos de máximo 3.

No fim devem submeter todas a mesma solução para ter a situação do aceite.
Depois de aceitação devem enviar-me um email com assunto e body definido em baixo. Se não fizer o sistema de detecção do plágio irá depois detectar uma situação do plagio e terão todas as submissões e avaliações anuladas.

```
Subject: Mooshak-C-Grupo
Body
Numero1
Numero2
Numero3
Jose Mfcfdfd
Artur Fcdfv
Maria Xdvdf
```

Linguagens e Bibliotecas

Pode usar código para estruturas de dados que acham necessários, como filas etc de qualquer fonte. Inserir referencia como comentário no seu código. Pode desenvolver a solução em vários ficheiros (.h...c etc) e depois juntar tudo em apenas um ficheiro para submeter ao Mooshak. Pode ser útil usar o standard template library do c++ e API da Java. Portanto adicionado compilação com c++, java e ocaml.
Discute a solução com os colegas e faça um plano do seu programa.

Um bom plano vale horas de programação