EXAMEN de CONTROLE CONTINU

Exercice 1 : QUESTIONS DE COURS (5 pts - 25 minutes)

Répondez chaque partie de la question :

1.1 Qu'est-ce un ***Système d'Exploitation*** (**SE**) ? A quoi sert-il ? Comment définir un bon **SE** ?

1.2 Qu'est-ce que la ***multiprogrammation*** ? Citer deux techniques de ***multiprogrammation*** et comment améliorent-elles le SE ?

1.3 Qu'est-ce le ***temps partagé*** ? Quelles sont ses avantages et inconvénients ?

1.4 Comment le **SE** sait quel partie de son code exécuter ? Citer 2 façons de communiquer avec le **SE**.

1.5 Citer 2 structures de données du **SE**. Que contiennent-elles ? A quoi servent-elles ?

Bonus : Dessinez l'automate fini de processus utilisateurs en ***temps partagé*** en tenant compte de la ***synchronisation***.

Exercice 2 : SYNCHRONISATION (5 pts - 25 minutes)

Dans un lavomatique, on cherche une solution pour permettre de répartir les machines à laver équitablement entre les clients. Considérez le programme suivant. Pour obtenir une machine, chaque client doit utiliser la fonction `allouer()`. Après usage de la machine, il doit utiliser `liberer()`.

Conditions initiales

```
#define NMACHINES 5
SEM * nlibre = CS(NMACHINES);
int dispo[NMACHINES] ={1,1,1,1,1};
```

Fonctions d'un Client

```
int allouer(){
      int i;
      P(nlibre);
      for (i=0; i < NMACHINES; i++)
            if (dispo[i] != 0) {
                  dispo[i] = 0;
                  return i;
            }
}

void liberer(int machine) {
      dispo[machine] = 1;
      V(nlibre);
 }
```

2.1 Quelles sont les ressources critiques ? A quoi sert le sémaphore `nlibre` ?
2.2 Ce programme respect-il les conditions de synchronisation ? Pourquoi ?
2.3 Corriger cette solution en utilisant le(s) sémaphore(s) nécessaire(s).

Exercice 3 : ORDONNANCEMENT (4 pts - 20 minutes)

Sur un ordinateur, l'***Ordonnanceur*** gère l'ordonnancement des processus par un tourniquet avec un ***quantum*** de 100 ms.

3.1 Soient les tâches ci-dessous, représentez dans un tableau l'évolution des tâches sur le processeur indiquant le **temps,** l'**événement** (***commutation, E/S, fin E/S***), l'**état de la file avant l'élection** ***(p:prêt, b:bloqué, x:en exécution),*** la **tâche élue** ainsi que **les dates de fin** pour chaque tâche.

Processus	T1	T2	T3	T4
Date d'arrivée	0	30	0	150
Durée	200	30	300	20
E/S		À 20ms, durée de 150ms		À 10ms, durée de 200ms

Exercice 4 : CRÉATION DES PROCESSUS (5 pts - 25 minutes)

Soit le programme suivant :

```
int main () {
  int i,p;
  for (i=4;i > 0;i=i/2) {
    p=fork();
    if (p==0) {
        printf("fils=%d\n",i);
    } else {
        printf("pere=%d\n",i);
    }
  }
  return (0);
}
```

4.1 Combien de processus vont êtres crées?
4.2 Quel est l'affichage produit (ne pas tenir compte de l'ordonnancement des processus)?
4.3 Modifier le programme de façon à ce que chaque père attende la fin de ses fils avant de se terminer, et que le père de tous les processus termine en affichant **« c'est fini ».**