Exercice 1 :
Soit le programme C suivant :

```
#include <stdio.h>
#include <unistd.h>

int main(void){
   int pid;

   pid = fork();

   if (pid > 0)
      printf("processus père: %d-%d-%d\n", pid, getpid(), getppid());

   if (pid == 0) {
      printf("processus fils: %d-%d-%d\n", pid, getpid(), getppid());
   }

   if (pid < 0)
      printf("Probleme de creation par fork()\n");

   return 0;
}
```

Saisir ce programme sous Linux. Soit *pgme1.c* le nom de ce programme. Lancez la compilation ensuite l'exécution de ce programme comme suit :
Compilation :
cc pgme1.c –o pgme1

Exécution :
./pgme1

Question :
Expliquez les informations qui s'affichent lors de l'exécution de ce programme ? Notez que la commande ***echo $$*** affiche le PID de la session (Shell) ouverte. Utilisez la commande ps –l pour vérifier ces informations.

Exercice 2 :

```
#include <stdio.h>
int m=2;
void main(void) {
  int i, pid;
  printf("m=%d\n", m);
  pid = fork();
  if (pid > 0) {
      sleep(1);          (a)
      m++;
      printf("\nje suis le processus père: %d, m=%d\n", i, m);
```

```
  }
  if (pid == 0) {
       sleep(1);                    (b)
       m = m*2;
       printf("\nje suis le processus fils: %d, m=%d\n", i, m);

  }
  if (pid < 0)  printf("Probleme de creation par fork()\n");

}
```

Question :
Exécutez le programme ci-dessus d'abord sans l'instruction (a) ensuite sans l'instruction (b). Expliquez les résultats obtenus dans les deux cas.

Exercice 3 :
Ecrire un programme qui crée deux fils : un qui calcule la somme de deux nombres a et b et l'autre qui calcule la soustraction de deux nombres c et d. Le père devra attendre la fin de ses fils à l'aide de la primitive *wait*() afin qu'il puisse calculer le produit des deux valeurs obtenues par les deux fils.