1. Il est assez courant d'échanger les valeurs de deux variables. On aimerait créer une fonction **swapf**, dont le prototype est donné ci-dessous, qui permette d'échanger le contenu de deux variables flottantes.

   ```
   void swapf(float * f1, float * f2);
   ```

   Implémentez la fonction **swapf**. Pour vous aider, voici comment on suppose que la fonction **swapf** sera appelée :

   ```
   void some_function()
   {
       float a = 4.2f;
       float b = 3.7f;

       swapf(&a, &b);
       // Maintenant, a vaut 3.7f et b vaut 4.2f.
   }
   ```

2. Implémentez la fonction **decouper**, dont le prototype est donné ci-dessous.

   ```
   void decouper(unsigned int duree, int * j,
                 int * h, int * m, int * s);
   ```

   Cette fonction reçoit une durée (en secondes) en entrée et doit découper cette durée en jours (dans **j**), heures (dans **h**), minutes (dans **m**) et en secondes (**s**). Ce découpage doit être tel que $duree = j * 24 * 60^2 + h * 60^2 + m * 60 + s$, $0 \leq h \leq 23$, $0 \leq m \leq 59$, $0 \leq s \leq 59$.

3. Soit p un polynôme de la forme $p(x) = ax^2 + bx + c$ avec $a \neq 0$. Les racines réelles de p sont les valeurs réelles de x telles que $p(x) = 0$. Soit $\Delta = b^2 - 4ac$ le déterminant de p.
   — Si $\Delta > 0$, p a deux racines $r1 = \frac{-b-\sqrt{\Delta}}{2a}$ et $r2 = \frac{-b+\sqrt{\Delta}}{2a}$.
   — Si $\Delta = 0$, p a une seule racine $r1 = \frac{-b}{2a}$.
   — Si $\Delta < 0$, p ne possède pas de racine **réelle**.
   Implémentez la fonction **racines**, dont le prototype est donné ci-dessous.

   ```
   int racines(double a, double b, double c,
               double * r1, double * r2);
   ```

   Cette fonction doit renvoyer le nombre de racines réelles du polynôme du second degré qu'elle reçoit en paramètres. Les paramètres **r1** et **r2** permettent de renvoyer les racines du polynôme. Vous pouvez vous servir de la fonction suivante pour calculer une racine carrée :

   ```
   #include <math.h>
   double sqrt(double x);
   ```

4. Que fait le code suivant ?

```
unsigned int unknown(unsigned int a, unsigned int b)
{
    unsigned int tab[a];
    unsigned int * p = NULL;
    unsigned int * p2 = &b;

    for (unsigned int i = 0; i < a; ++i)
        *(tab+i) = *(&b);

    *p2 = 0;

    for (p = tab + a - 1, *p2=0; p >= tab; --p)
        *p2 = *p + b;

    return *p2;
}
```

5. Soit **Job** la stucture suivante :

```
typedef struct
{
    int job_id;
    float runtime;
} Job;
```

Créez les fonctions suivantes :

```
Job * allocate_job(int job_id, float runtime);
void deallocate_job(Job * job);
void print_job(const Job * job);
```

Telles que :
— **allocate_job** alloue un nouveau Job, affecte ses champs correctement en fonction des paramètres d'entrée de la fonction puis renvoie le nouveau Job créé.
— **deallocate_job** libère la mémoire occupée par un Job.
— **print_job** affiche ce qu'est un job (tous ses champs).