```c
#include "utils.h"

int calculNbTour(int tailleCircuit){
  int res = NB_KM / tailleCircuit;
  return res;
}

int msleep(long tms){
  struct timespec ts;
  int ret;

  if (tms < 0)
  {
    errno = EINVAL;
    return -1;
  }

  ts.tv_sec = tms / 1000;
  ts.tv_nsec = (tms % 1000) * 1000000;

  do {
    ret = nanosleep(&ts, &ts);
  } while (ret && errno == EINTR);

  return ret;
}

void swap(struct Voiture *x, struct Voiture *y){
  struct Voiture temp = *x;
  *x = *y;
  *y = temp;
}

void triVoiture(struct Voiture *voitures, int choix){
  for (int i = 0; i < 19; i++){
    for (int j = 0; j < 19 - i; j++){
      if(choix == 5){
        if (voitures[j].nbTour < voitures[j+1].nbTour){
          swap(&voitures[j], &voitures[j+1]);
        }
      }
      else{
        if (voitures[j].best[choix] > voitures[j+1].best[choix]){
          swap(&voitures[j], &voitures[j+1]);
        }
      }
    }
  }
}

//Renvoie un entier random entre min et min + max
double randomInt(int max, int min){
  return (rand() % max) + min;
}

//Renvoie un temps double au centiÃ¨me
double randomTime(int min, int max){
  double pEntiere = randomInt(min, max);
  double pDecimale = randomInt(999, 0);
  pDecimale /= 1000;
  return pEntiere + pDecimale;
}

double calculSection(){
  return randomTime(15, 35);
}
```

```
double meilleurTemps(double t1, double t2){
  double meilleur;
  if (t1 < t2) {
    meilleur = t1;
  }
  else{
    meilleur = t2;
  }
  return meilleur;
}

double stand(){
  double tempsAdditionnel = 0.0;
  if (randomInt(100,0) >= 97) {
    tempsAdditionnel = randomTime(10, 10);
  }
  return tempsAdditionnel;
}

int out(){
  int resultat = 0;
  if (randomInt(100,0) >= 99) {
    resultat = 1;
  }
  return resultat;
}

void initVarVoiture(struct Voiture *v){
  for(int i = 0; i < 3; i++){
    v->best[i] = 51; //secondes par section
  }
  v->best[3] = 151;
  v->pit = 0;
  v->out = 0;
}

struct Voiture init_voiture(int i){
  int numVoitures[20] = {7,99,5,16,8,20,4,55,10,26,44,77,11,18,23,33,3,27,6,85};
  struct Voiture v;
  struct Voiture *pointeurVoiture = &v;
  v.numero = numVoitures[i];
  initVarVoiture(pointeurVoiture);
  return v;
}

void docRecap(struct MemoirePartagee *tabVoitures, int choix){
    FILE *fichier = NULL;
    switch (choix) {
        case 0:
            fichier = fopen("RecapEssai1.txt", "a");
            break;
 case 3:
            fichier = fopen("RecapEssai2.txt", "a");
            break;
        case 1:
            fichier = fopen("RecapEssai3.txt", "a");
            break;
        case 20:
            fichier = fopen("RecapQualifications1.txt", "a");
            break;
        case 15:
            fichier = fopen("RecapQualifications2.txt", "a");
            break;
        case 10:
```

```
            fichier = fopen("RecapQualifications3.txt", "a");
            break;
        case 2:
            fichier = fopen("RecapCourseFinal.txt", "a");
            break;
    }
    if(fichier != NULL){
        struct MemoirePartagee mem = *tabVoitures;
        struct MemoirePartagee *pointeurMem = &mem;
        int finale = 0;
        if(choix == 2){
          triVoiture(pointeurMem->tableauV, 5);
          finale = 1;
        }
        else{
          triVoiture(pointeurMem->tableauV, 3);
        }
        system("clear");
        fprintf(fichier,"_____\n");
        switch (choix) {
            case 0:
                fprintf(fichier,"|            ESSAIS 1           |\n");
                break;
     case 3:
                fprintf(fichier,"|            ESSAIS 2           |\n");
                break;
            case 1:
                fprintf(fichier,"|            ESSAIS 3           |\n");
                break;
            case 20:
                fprintf(fichier,"|         QUALIFICATIONS 1        |\n");
                break;
            case 15:
                fprintf(fichier,"|         QUALIFICATIONS 2        |\n");
                break;
            case 10:
                fprintf(fichier,"|         QUALIFICATIONS 3        |\n");
                break;
            case 2:
                fprintf(fichier,"|          COURSE FINALE          |\n");
                break;
        }
        if (choix < 4) {
            choix = 20;
        }
        fprintf(fichier,"|_____|\n");
        fprintf(fichier,"| Voiture |   Tour    | Pit | Out |\n");
        fprintf(fichier,"|---------|-----------|-----|-----|\n");
        if(finale == 1){
          for(int i = 0; i < choix; i++){
            fprintf(fichier,"|   %2d    |   %2.0f     |  %d  |  %d  |\n", mem.tableauV[i].numero, (mem.tableauV[i].nbTour),
mem.tableauV[i].pit, mem.tableauV[i].out);
            fprintf(fichier,"---------------------------------|\n");
          }
        }
        else{
          for(int i = 0; i < choix; i++){
            fprintf(fichier,"|   %2d    |   %.3f\"  |  %d |  %d |\n", mem.tableauV[i].numero, mem.tableauV[i].best[3],
mem.tableauV[i].pit, mem.tableauV[i].out);
            fprintf(fichier,"|---------------------------------|\n");
          }
        }
        fprintf(fichier,"|  Best   | Section 1 |\n");
        fprintf(fichier,"|---------|-----------|\n");
```

```
        triVoiture(pointeurMem->tableauV, 0);
        fprintf(fichier,"|   %2d    |   %.3f\" |\n", mem.tableauV[0].numero, mem.tableauV[0].best[0]);
        fprintf(fichier,"|---------|----------|\n");
        fprintf(fichier,"|  Best   | Section 2 |\n");
        fprintf(fichier,"|---------|----------|\n");
        triVoiture(pointeurMem->tableauV, 1);
        fprintf(fichier,"|   %2d    |   %.3f\" |\n", mem.tableauV[0].numero, mem.tableauV[0].best[1]);
        fprintf(fichier,"|---------|----------|\n");
        fprintf(fichier,"|  Best   | Section 3 |\n");
        fprintf(fichier,"|---------|----------|\n");
        triVoiture(pointeurMem->tableauV, 2);
        fprintf(fichier,"|   %2d    |   %.3f\" |\n", mem.tableauV[0].numero, mem.tableauV[0].best[2]);
        fprintf(fichier,"|--------------------|\n");
        fclose(fichier);
    }
    else{
        printf("Ouverture du fichier recap impossible");
    }
}
```